A Report

ON

Wireless Video Streaming in Ad-Hoc Mesh Networks

BY

Kartic Bhargav K.R

Under the Supervision of

Dr. Bharadwaj Amrutur

At

Indian Institute of Science, Bangalore

In collaboration with

Robert Bosch Centre for Cyber Physical Systems, Bangalore

March 2013

Table of Contents

Ad-Hoc Networks – A Literature Review

Ad-Hoc Network - Basics

An ad hoc wireless network is a collection of wireless nodes that self-configure to form a network without the aid of any established infrastructure. Some or possibly all of these nodes could be mobile. These networks are extremely compelling for applications where a communications infrastructure is too expensive to deploy, cannot be deployed quickly, or is simply not feasible. There are numerous potential applications for ad hoc wireless networks, ranging from multihop wireless broadband Internet access, to sensor networks, to building or highway automation, to voice, image, and video communication for disaster areas.

Challenges & Limitations

The lack of established infrastructure, the network and channel dynamics, and the nature of the wireless medium offer an unprecedented set of challenges in supporting demanding applications over ad hoc wireless networks. The wireless channel is inherently a broadcast medium, so transmissions from different nodes interfere with each other. The quality of wireless links varies over time and space due to interference, multipath fading, and shadowing. Network conditions are highly dynamic as nodes join and leave the network in an unpredictable manner. Furthermore, as new links are formed and others vanish, routing of traffic from one node to another may frequently change. These daunting challenges have spurred a large body of research on the design, analysis, and fundamental performance limits of ad hoc wireless networks.

Video Streaming via Ad-Hoc Networks - Requirements

For video streaming, high bandwidth requirements are coupled with tight delay constraints as packets need to be delivered in a timely fashion to guarantee continuous media playout. When packets are lost or arrive late, the picture quality suffers as decoding errors tend to propagate to subsequent portions of the video. Due to the high bit rate requirements of video, a media stream may congest the network significantly. Hence, it is imperative to account for the potential impact of each video user on the network statistics and guarantee that the network is not operating beyond its capacity.

Unfortunately, most network designs do not provide mechanisms for protocol layers to optimally adapt to underlying channel conditions and specific application requirements. While protocol layering is an important abstraction that reduces network design complexity, it is not well suited to wireless networks since the nature of the wireless medium makes it difficult to decouple the layers. Moreover, meeting the end-to-end performance requirements of demanding applications is extremely challenging without interaction between protocol layers.

It is believed that video streaming over ad hoc wireless networks can benefit substantially from a cross-layer design. In this design, interdependencies between layers are characterized and exploited by adapting to information exchanged between layers and building the appropriate amount of robustness into each layer. For example, routing protocols can avoid links experiencing deep fades, or the application layer can adapt its transmission rate based on the underlying network throughput and latency.

Link capacities are dynamically reallocated based on link state information, and stream based multipath source routing and scheduling is performed by balancing network congestion and distortion of real-time media streams. This framework includes new techniques to jointly optimize error-resilient source coding, packet scheduling, stream-based routing, link capacity assignment, and adaptive link layer techniques. The optimization is carried out dynamically by all nodes, based on link state communication, to continuously adapt to the changing wireless link conditions and traffic flows.

Supporting multimedia applications over wireless links has been one of the main fields of attention in the networking and video coding communities in the last decade. For such applications, it is well known that the separation principle of source and channel coding put forth in Shannon's information theoretic framework does not hold. Hence, joint source and channel coding techniques have been proposed to overcome the challenges of wireless links. However, these have not yet led to a unified solution to this problem. Nevertheless, these considerations have strongly influenced the video community in the design of the new H.264 video coding standard, which incorporates a highly flexible syntax well suited to network transmission.

As the number of nodes of a wireless network grows, interference increases, reducing the achievable data rates. The capacity of a static wireless ad hoc network is shown to asymptotically vanish as the number of users increases. However, recent results show that in more practical settings, the high data rates and low delay constraints of multimedia applications may be supported. The 802.11 protocol operating in ad hoc mode provides an interesting benchmark against which proposed designs for ad hoc networks may compete. In its simplest form, it allows for one user at a time to be active within a carrier sense region (typically on the order of a few hundred meters). As only active hosts occupy the wireless medium, this medium access control (MAC) protocol comes closer to the achievable capacity than time-division multiple access (TDMA) or frequency-division multiple access (FDMA) systems.

Performance in a Multi-Hop Setup: Loss of Bandwidth by 50% after every hop.

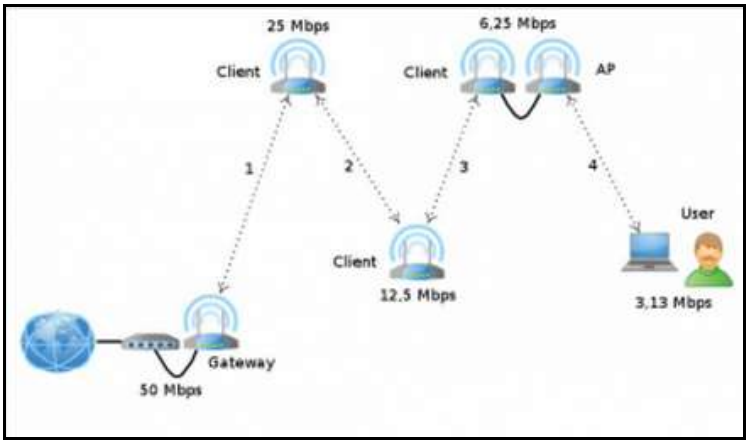

Figure: Bandwidth Analysis in a Multi Hop Setup

The relationship between performance, bandwidth and hop is something to take into account when seeking to leverage the available resources and maximize bandwidth as much as possible for each hop, the wireless bandwidth is halved.

More the number of jumps to the final destination, lower is the bandwidth between the sender and the receiver. In order to reduce the overhead the setup may have to manually manipulate routes or establish connections and replace wireless connections by cable.

Since after every hop, we cut bandwidth in half, there could arise a point after which Layer 3 protocols will be unreachable or unusable.

Related Work – Wireless Video Streaming via BeagleBoard

In an earlier related project, Wireless Video Streaming along with Locational Identification has been implemented for a single-hop on a BeagleBoard.

The work in this Project involved the development of a mobile locationing system using a Wireless Network. This system was also capable of estimating the position of a person and streaming video over the wireless network. This system could also be used inside a building (Indoors) without the explicit need of any external infrastructure inside the building. The system uses multiple sensors like Camera and Inertial Measurement Units. All these modules were connected to the BeagleBoard-xM.

The BeagleBoard-Xm collected data from these sensors, processed it and then transmitted it to the local station over the wireless network. The camera could be placed inside the Helmet and the Inertial Measurement Unit (IMU-OpenShoe) was placed inside the shoe of the person whose position was to be estimated. The video stream from camera was compressed on the BeagleBoard-Xm DSP and transmitted over the wireless network to the local host. The IMU was based on the work done under the Project OpenShoe. With the help of IMU we get the position and velocity estimates, which were then transmitted over Wireless Network to the local station. The main aim of the project was to build a working system that will have the camera video codec running on the DSP Processor of BeagleBoard-Xm and IMU. Both these worked together and transmitted the compressed video stream along with the IMU data (Position and Velocity estimates) to the local station.

At the local station, the position data was extracted from the received data format and the position of the person was plotted on GNU plot. IMU data streaming over wireless network was done by TCP/IP socket programming. Development of the Project First Responder System involved estimation of position and video streaming in an indoor environment. The system was integrated with the BeagleBoard-Xm, a Video Capture Device, IMU and wireless connectivity modules. Video Compression was done on the BeagleBoard itself. The IMU gave the position and velocity details of the FirstResponder. BeagleBoard-Xm collected data from IMU and the video capture device, processed it and transmitted it over the Wireless Network.

Raspberry Pi – Salient Features

The Raspberry Pi is a credit-card sized computer that plugs into a TV and a keyboard. It's a capable little PC which can be used for many of the things that the desktop PC does, like spreadsheets, word-processing and games. It also plays high-definition video.

The Raspberry Pi has a Broadcom BCM2835 system on a chip (SoC), which includes an ARM1176JZF-S 700 MHz processor (The firmware includes a number of "Turbo" modes so that the user can attempt overclocking, up to 1 GHz, without affecting the warranty), and a VideoCore IV GPU. The ARM11 introduced the ARMv6 architectural additions. It does not include a built-in hard disk or solid-state drive, but uses an SD card for booting and long-term storage.

Figure: The Raspberry Pi

The Raspberry Pi comes with a variety of Operating Systems each at its own level of advancement and convenience. Typically, after choosing the operating system, the image is burned on to a freshly formatted SD Card. To use an image file, one will need to unzip it and write it to a suitable (2GB or larger, 4GB or larger for Raspbian) SD card using the UNIX tool dd. Windows users should use the Win32DiskImager. Dragging and dropping/ copying would not work in any of these cases.

For the first-time Pi users it is recommended that the New out of Box Software (NOOBS) is installed onto a 4GB (or larger) SD card. On first boot, it presents us with a choice of operating systems to install, including Raspbian, Pidora and two flavors of XBMC. Once we have installed an operating system, we can return to the NOOBS interface by holding down shift during boot; this allows to switch to a different operating system, or overwrite a corrupted card with a fresh install of the current one.

By default, NOOBS will output over HDMI at the display's preferred resolution, even if no HDMI display is connected. If you do not see any output on your HDMI display or are using the composite output, press 1, 2, 3 or 4 on your keyboard to select HDMI preferred mode, HDMI safe mode, composite PAL mode or composite NTSC mode respectively.

Once through with this, one finds that the Raspbian Wheezy is very user friendly and has features in similar lines to a Fedora/ Ubuntu Operating System. Raspbian is a free operating system based on Debian optimized for the Raspberry Pi hardware. An operating system is the set of basic programs and utilities that make the Raspberry Pi run. Raspbian comes with over 35,000 packages, pre-compiled and optimized for best performance on the Raspberry Pi.

Once the Raspbian Wheezy raw image is booted from an SD Card, the following picture shows how to quickly setup the Pi.

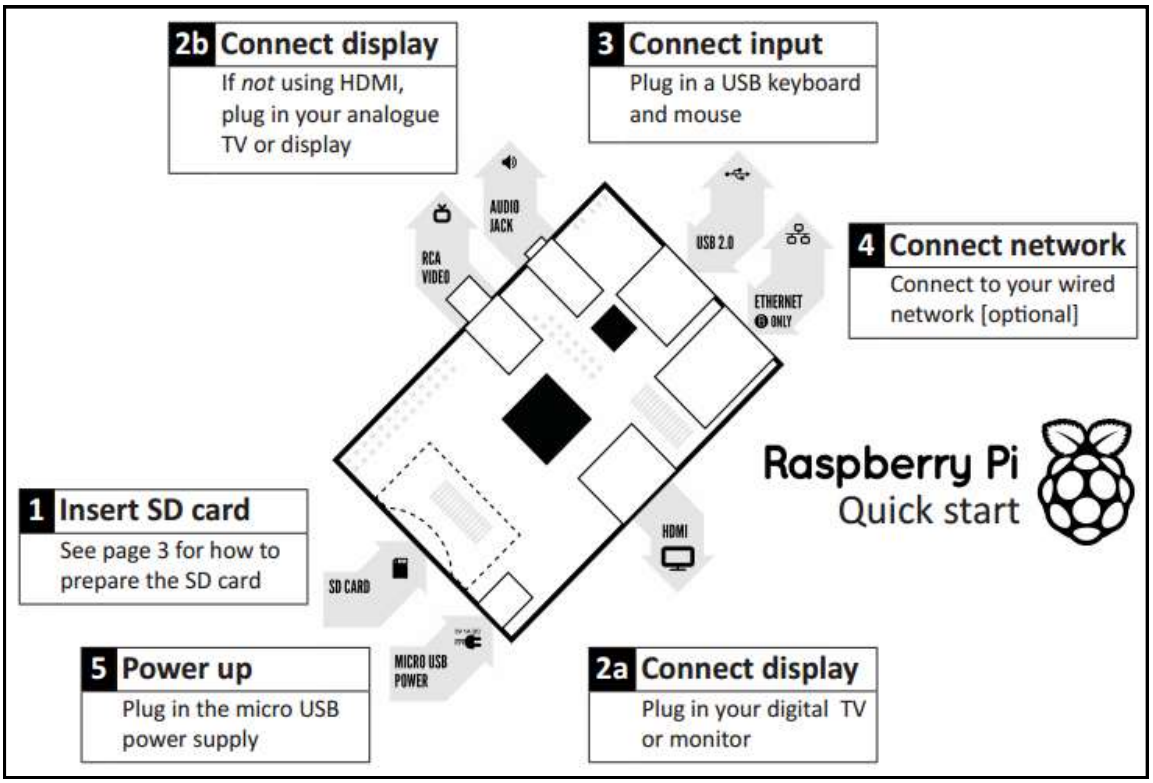

Figure: Raspberry Pi Quick Setup

On first boot one comes to the Raspi-config window where setting such as timezone and locale can be changed to suit the operator. Once the required settings have been modified, expand the filesystem (by expand_rootfs) and reboot the Pi. Typing ‘startx’ in the prompt, loads the raspberry Pi desktop.

Routing in Ad-Hoc Mesh Networks

A wireless mesh network is a fully wireless network that employs multihop communications to forward traffic en route to and from wired Internet entry points. Different from flat ad hoc networks, a mesh network introduces a hierarchy in the network architecture with the implementation of dedicated nodes (called wireless routers) communicating among each other and providing wireless transport services to data traveling from users to either other users or access points (access points are special wireless routers with a high-bandwidth wired connection to the Internet backbone). The network of wireless routers forms a wireless backbone (tightly integrated into the mesh network), which provides multihop connectivity between nomadic users and wired gateways. The meshing among wireless routers and access points creates a wireless backhaul communication system, which provides each mobile user with a low-cost, high-bandwidth, and seamless multihop interconnection service with a limited number of Internet entry points and with other wireless mobile users.

Mesh networks are built on a mix of fixed and mobile nodes interconnected via wireless links to form a multihop ad hoc network. As in MANETs, users' devices are an active part of the mesh. They dynamically join the network, acting as both user terminals and routers for other devices, consequently further extending network coverage. Mesh networks thus inherit many results from MANET research but have civilian applications as the main target.

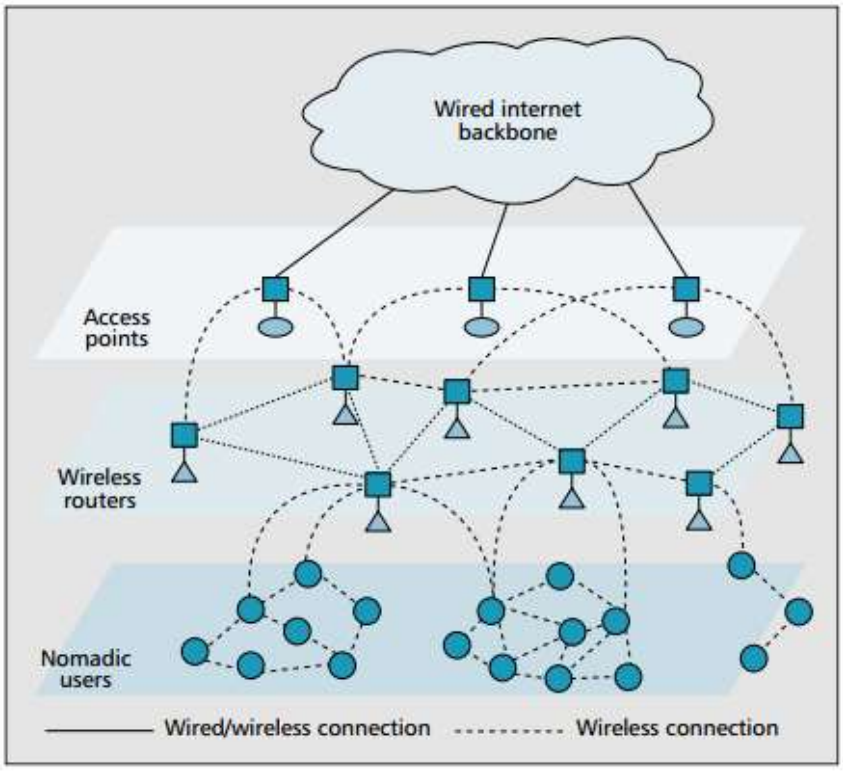

Figure: A 3 tier Architecture for Ad-Hoc Mesh Network

Few years back, the IEEE 802 creating Task Group 802.11s aimed at defining medium access control (MAC) and PHY layers for mesh networks to improve wireless LAN (WLAN) coverage with no single point of failure. In such networks, 802.11 access points relay information from one to another, hop by hop, in router-like fashion. As users and access points are added, capacity is added.

Under the hood of a mesh there is no default gateway, there are only neighboring routers that will relay traffic. Also, due to the sheer number of routers in a mesh we don't have to worry about your active connections dropping just because you walked three blocks away and your smartphone decided that another two or three mesh nodes were its preferred ("default") "gateways"; this is called mobile IP.

Optimised link state routing (OLSR) Protocol

OLSR is a point-to-point routing protocol based on the traditional link-state algorithm. In this strategy, each node maintains topology information about the network by periodically exchanging link-state messages. The novelty of OLSR is that it minimizes the size of each control message and the number of rebroadcasting nodes during each route update by employing multipoint replaying (MPR) strategy. To do this, during each topology update, each node in the network selects a set of neighbouring nodes to retransmit its packets.

This set of nodes is called the multipoint relays of that node. Any node which is not in the set can read and process each packet but do not retransmit. To select the MPRs, each node periodically broadcasts a list of its one hop neighbours using hello messages. From the list of nodes in the hello messages, each node selects a subset of one hop neighbours, which covers all of its two hop neighbours.

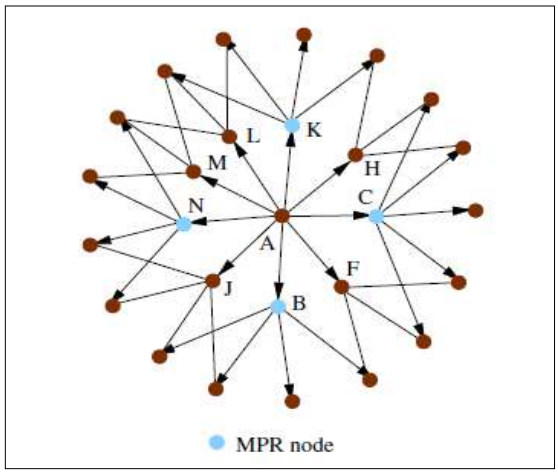

Figure: Multipoint Relays (MPR)

For example, in Fig., node A can select nodes B, C, K and N to be the MPR nodes. Since these nodes cover all the nodes, which are two hops away. Each node determines an optimal route (in terms of hops) to every known destination using its topology information (from the topology table and neighbouring table), and stores this information in a routing table. Therefore, routes to every destination are immediately available when data transmission begins.

B.A.T.M.A.N. (Better Approach To Mobile Adhoc Networking)

The Better Approach To Mobile Adhoc Networking (B.A.T.M.A.N.) is a routing protocol for multi-hop ad hoc mesh networks which is under development and intended to replace OLSR.

B.A.T.M.A.N.'s crucial point is the decentralization of the knowledge about the best route through the network — no single node has all the data. This technique eliminates the need to spread information concerning network changes to every node in the network. The individual node only saves information about the "direction" it received data from and sends its data accordingly. Hereby the data gets passed on from node to node and packets get individual, dynamically created routes. A network of collective intelligence is created.

In early 2007 the batman developers started experimenting with the idea of routing on layer 2 (Ethernet layer) instead of layer 3. To differentiate from the layer 3 routing daemon the suffix "adv" (advanced) was chosen. Instead of sending UDP packets and manipulating routing tables, it provides a virtual network interface and transparently transports packets on its own. The batman-adv kernel module is part of the official Linux kernel since 2.6.38.

B.A.T.M.A.N. does have elements of classical routing protocols: It detects other B.A.T.M.A.N. nodes and finds the best way (route) to these. It also keeps track of new nodes and informs its neighbours about their existence.

In static networks, network administrators or technicians decide which computer is reached via which way or cable. As radio networks undergo constant changes and low participation this task has to be automated as far as possible.

On a regular basis, every node sends out a so-called "broadcast" (a general message to all) thereby informing all its neighbours about its existence. The neighbors then relay this message to their neighbours and so on and so forth. This carries the information to every node in the network. In order to find the best way to a certain node, B.A.T.M.A.N counts the originator-messages received and logs which neighbour the message came in through.

Like distance-vector protocols, but unlike link-state protocols, B.A.T.M.A.N does not try to determine the whole way, but, by using the originator-messages, only the package's first step in the right direction. The data is handed over to the next neighbour in that direction, who in turn uses the same mechanism. This process is repeated until the data reaches its destination.

Besides for radio networks, B.A.T.M.A.N can also be used with common cable connections, such as Ethernet.

Developments in the Latest Version of B.A.T.M.A.N – Version 3

The greatest innovation in the latest version is B.A.T.M.A.N.'s support of multiple network devices. Now a computer or router running B.A.T.M.A.N can be deployed on a central point, like a church or another high building, and have several wired or wireless network interfaces attached to it. When so deployed, B.A.T.M.A.N can relay network data in more than one direction without any retransmission delay.

Certain unusual phenomena and special circumstances could appear during the determination of the best route through the network. These have been tackled and counteracted to prevent circular routing (which can prevent data reaching its destination) from occurring.

A node can now inform the network that it provides access to the Internet. Other nodes use that information to evaluate whether there is a connection to the Internet close to them and what bandwidth is available. They can

either use a specific gateway or allow B.A.T.M.A.N to determine which gateway to use, based on criteria such as connection speed.

Announcing devices not running B.A.T.M.A.N themselves was also included in this version. Usually this method is used to connect house-networks to mesh-networks. An antenna installation on the roof will connect to the wireless network through B.A.T.M.A.N and the rest of the house will simply be announced thus also be reachable.

This version of B.A.T.M.A.N. has been shown to exhibit high levels of stability but slightly slow convergence times in real-world conditions; this is confirmed by theoretical analysis.

Layer 2?

Most other wireless routing protocol implementations (e.g. the Batman Daemon or BatmanD) operate on layer 3 which means they exchange routing information by sending UDP packets and bring their routing decision into effect by manipulating the kernel routing table.

Batman-adv operates entirely on OSI Layer 2 - not only the routing information is transported using raw ethernet frames but also the data traffic is handled by batman-adv. It encapsulates and forwards all traffic until it reaches the destination, hence emulating a virtual network switch of all nodes participating. Therefore all nodes appear to be link local and are unaware of the network's topology as well as unaffected by any network changes.

This design bears some interesting characteristics:

- Network-layer agnostic - you can run whatever you wish on top of batman-adv: IPv4, IPv6, DHCP, IPX.
- Nodes can participate in a mesh without having an IP
- Easy integration of non-mesh (mobile) clients (no manual HNA fiddling required)
- Roaming of non-mesh clients
- Optimizing the data flow through the mesh (e.g. interface alternating, multicast, forward error correction, etc)
- Running protocols relying on broadcast/multicast over the mesh and non-mesh clients (Windows neighborhood, mDNS, streaming, etc)

Batctl ?

To still have a handy tool to configure & debug the batman-adv kernel module, the batctl tool was developed. It offers a convenient interface to the entire module's settings as well as status information. It also contains a layer 2 version of ping, traceroute and tcpdump, since the virtual network switch is completely transparent for all protocols above layer 2.

Batman-adv encapsulates and forwards all traffic until it reaches the destination, hence emulating the virtual network switch of all Participating nodes. Therefore all nodes appear to be local link and are unaware of the network's topology as well as network unaffected by any changes.

Zero-Config with Batman-adv

One of the advantages of Batman-adv is what is called the Zeroconfig. In other words it is the ability of setting up a mesh network without having to configure any software parameters whatsoever after the firmware is installed in a router.

Let us make a distinction about a few important possible routing scenarios at this point.

1. **Scenario 1 (Layer3)**

In the usual OLSR mesh; one sets up the router interfaces in Ad Hoc mode; gives the interface a Net range, an IP address and then the router is ready to work. All the clients of that routing node (whether they are routing devices or users) will connect to the node in the Ad Hoc mode. Packets are forward on layer 3 as well as routing and through Ad Hoc (at the Data Link Layer level).

In the Raspberry Pi, a proper configuration would give us one wireless interface which would work in Ad Hoc mode. Clients (users) and client nodes will use the same SSID, network setup and share everything in common with the routing node. One of the disadvantages of this type of setup is that everyone will be connecting through Ad Hoc and via the same channel, the same radio and so on, thereby reducing and limiting many more functional aspects of the mesh.

In this typical scenario our routing device has only one wireless radio that will bare all the network activity and in order to maximize your network potential, increase security and functionality you will move to scenario 2.

- **Scenario 2 (Layer3)**

Everything as described above stays with only one change - we will now use 2 routing devices.

One routing device will work only in the {Ad Hoc mode + OLSR} and will have its specific Network Mask, Range & Broadcast, SSID, Channel, etc. This routing device will work only to build the Mesh backbone and all the Mesh links. Only Mesh nodes will connect to it to carry the mesh traffic/packets (not external users).

The second routing device will be used ONLY by users to connect to the network and it will have its own different IP range; may be or may not be in the same network, will have a different SSID and channel etc.

Now, why do we want this? - To maximize functionality at all levels. But note that having a second routing device is not mandatory but it is a better setup in all possible ways.

While we are still on layer 3 the second routing device does not need to be a Physical device as long as the first routing device is capable of either of VAP (Virtual Access Point) which would allow it's radio chipset to create a new SSID using the same radio or it is also possible that the routing device could have 2 separate radios.

One radio creates the Mesh Backbone at only for routing devices and the other radio is the AP to enable user connectivity.

Still in these scenarios we could have both the backbone and the AP in the same Network range; divide the subnet or have 2 different ones bridged to each other.
The routing is always done in Ad Hoc and it is the OLSR that chooses the routes to use. All this works on layer 3. Ad Hoc by itself would not know what to do to forward the packets the best way which is why we need a

routing protocol and at layer 3 we will need to specify a Net range for the Ad Hoc to work properly.

- **Scenario 3 (Layer 2)**

If we are using one routing device only with one radio only you will have to create the Ad Hoc network but we will not need to give the Ad Hoc network any range or specifications.

Nothing is needed accept adding batman-adv on top of it to figure out what is the best way to forward packets and this is all done at level 2.

Unlike Ad hoc+OLSR or Ad hoc + BatmanD (A Layer 3 Routing Protocol by the B.A.T.M.A.N Developers that is no longer developed); we will not need to configure the network specifics but only to add bat0 to the LAN/eth0 (Zero Config).

We will then connect to eth0 (Wire) or wlan0 (Wireless) while bat0 is taking care of the routing using Ad hoc and the routing is done by batman-adv which will forward the packets the way it thinks it is best according to pre-set specifications and internal functioning.

- **Scenario 4 (Layer 2)**

We can add a second routing device just like the examples for layer 3 and in case we wish to have users connecting to the mesh - they will use the AP routing device which gets their packets forwarded by the Ad hoc routing device running batman-adv.

If our routing device has 2 radios or allows us to create a VAP, then the same applies to the Layer 3 example and in this case the only thing we need to do is to enslave the batman-adv interface to your layer 3 configuration without having to do anything else.

Batman-adv does not need to be configured like layer 3 protocols but will by itself manage the routing path and forward traffic as it thinks it's best based on a few rules we can specify for bat-adv to follow.

We will always work on Layer 3 to transmit data through the nodes but if we want to manipulate, routes, packets and traffic at Layer 2 then we can for example use ebtables to do so (just like we would use iptables for Layer 3).

Batman-adv Manual – Setting up the Mesh

Read the complete documentation including the side notes since each of this could possibly impact the network deployment scenario.

Components Needed

- Raspberry Pi (3) with Power Up cables & SD Cards (with fresh Raspbian image installed)
- Wireless Adaptors (min. 3)
- Category 5 Cable(s)
- Keyboard ; Mouse ; Display Monitor with HDMI input (direct/via external cable)

Once the Pi is set up in the usual configuration with the SD Card flashed with the fresh Raspbian image (As explained earlier in the Raspberry Pi Quick Setup), next step is to establish Network Connectivity since the latest compatible version of batman-adv has to be loaded on each of the boards.

We focus on connecting the Pi to the IISc LAN and thereby obtaining Internet Access to aid our cause. Note: If a direct connection to the Internet exists => this section can be skipped.

1. Establishing IISc LAN Connectivity & gaining Internet access

Perform the following directly at the Command Screen or by logging into Terminal after booting to desktop (by 'startx')

```
pi@raspberrypi ~ $ sudo nano /etc/resolv.conf

nameserver 127.0.0.1
nameserver 10.16.25.13     #This is to specify the DNS


pi@raspberrypi ~ $ sudo nano /etc/network/interfaces

#LoopBack Interface
auto lo
iface lo inet loopback

#Eth0 Interface
auto eth0
iface lo inet static #To assign a static IP. Possible Combns are
static/dhcp/manual
address 10.32.33.221 #Specify each Pi's IP Address as mentioned. (221-223)
netmask 255.255.255.0
gateway 10.32.33.1
network 10.32.33.0
broadcast 10.32.33.255

#wlan0 interface
allow-hotplug wlan0
iface wlan0 inet manual
```

```
pi@raspberrypi ~ $ sudo nano /etc/environment

http_proxy="http://proxy.serc.iisc.ernet.in:3128/"
https_proxy="https://proxy.serc.iisc.ernet.in:3128/"
ftp_proxy="ftp://proxy.serc.iisc.ernet.in:3128/"
socks_proxy="socks://proxy.serc.iisc.ernet.in:3128/"
```

```
pi@raspberrypi ~ $ sudo nano /etc/apt/apt.conf

Acquire::http::proxy "http://proxy.serc.iisc.ernet.in:3128/";
Acquire::https::proxy "https://proxy.serc.iisc.ernet.in:3128/";
Acquire::ftp::proxy "ftp://proxy.serc.iisc.ernet.in:3128/";
Acquire::socks::proxy "socks://proxy.serc.iisc.ernet.in:3128/";
```

```
#Restart/Reload Networking
pi@raspberrypi ~ $ sudo /etc/init.d/networking restart
pi@raspberrypi ~ $ sudo /etc/init.d/networking reload
```

Also plug in the necessary proxy details in the respective browser (if needed).

To check the LAN Connectivity, try pinging another Node in the IISc LAN (on L3 as of now). For example: < `ping 10.32.33.160` >.

2. Batman-adv Installation

Next we move to batman-adv installation. This can be coded into a single script file (bat-setup.sh) and run after the Pi is rebooted with Network Connectivity/Internet Access present.

Another option is to include the script file in the list of programs to be run during boot up.
{For this=> `update-rc-d bat-setup.sh enable` }

```
sudo nano bat-setup.sh
sudo modprobe batman-adv
sudo modprobe ebatables
sudo modprobe ipv6
sudo iw phy phy0 interface add ah0 type ibss
sudo iw phy phy0 set distance 20000 #Atheros only

sudo ifconfig ah0 mtu 1532
sudo iwconfig ah0 mode ad-hoc essid mymesh ap any channel 1
sudo ifconfig wlan0 down
sudo batctl if add ah0
sudo ifconfig ah0 up
sudo brctl addbr mesh0
sudo brctl addif mesh0 eth0
sudo brctl addif mesh0 bat0
sudo ifconfig eth0 up
sudo ifconfig bat0 up
sudo ifconfig mesh0 up
```

- Once this is done: `ifconfig` would show the updated set of interfaces.
- What this script file describes after installing the compatible batman-adv module is the addition of the Ad Hoc Interface to the wlan0. The wlan0 is what comes up as the phy0 physical interface.
- One important point to note before deploying the Wireless Adapter is to check if its wireless chipset supports Ad Hoc & Mesh connectivity. RaLink and Atheros are popular chipsets that do, but ones like RealTek don't.
- Hence, to test if your Wireless Adapter indeed has the appropriate inbuilt features, before setting up the mesh network: If a wlan0 pops up in `ifconfig` => the driver is present and compatible with the current version of Linux. But when it also shows up after `iw list` => The adapter also can support Mesh Networking (has the ability to go into the Ad Hoc mode).
- Another important point to keep in mind is that a card can support a feature that the driver that we use may not support. So do make sure the driver also does support the feature without issues.
- For the current network setup, the TP Link TL WN722N Wireless Adapter was used. This has an Atheros Chipset with Ad Hoc & Mesh capabilities and also has the ability to support the VAP mode.

3. Assigning the IP Address

Regarding the IP Address allocation during the running of the script file, we initially set an IP for the Layer 3 interface (eth0) as is done to establish Internet Access. Then we enslave your wireless interface (wlan0) to bat0 and create a bridge (mesh0) that will have eth0 and bat0 in it and

a) For a simple mesh network, we specify the IP in the bat0

b) While for a Mesh where non-batman systems can be added to every node, it is in this bridge (mesh0) that we specify the IP.

In the conventional OLSR Mesh Setup, there is a need to have either a split network range (where one interface uses part of the range and the other uses the rest of the range) or each interface uses a different IP class. (This is one of the reasons why many have dropped the OLSR mesh and moved to easy-to-deploy batman-adv instead). So, our current setup being entirely based on a Layer 2 Mesh Network, is much different. All we need to have is one IP per 2 physical interfaces which will be eth0 and the wireless interface (bat0).

Hence depending upon the features expected of the mesh node, the interface to assign the IP accordingly differs. Setting up the IP in one of the 2 possible options:-

a) **Simple Mesh Network Setup**

Here a node can only possess routing / relaying functions within the batman-adv network. It is hence exclusively a bat node which cannot act as an AP or a gateway.

```
pi@raspberrypi ~ $ sudo nano /etc/network/interfaces

auto lo
iface lo inet loopback

auto eth0
iface eth0 inet manual


auto bat0
iface bat0 inet static
address 192.168.2.100
netmask 255.255.255.0
#gateway 192.168.2.1


#allow-hotplug wlan0
#iface wlan0 inet manual
sudo /etc/init.d/networking restart #Restart Networking
```

Note: The gateway setting is optional and will be used when one of the nodes is configured to be a gateway (discussed later).

Once the Networking is restarted, the `ifconfig` should read similar to the following:

```
pi@raspberrypi ~ $ ifconfig
ah0       Link encap:Ethernet  HWaddr 10:fe:ed:12:60:ea
          inet6 addr: fe80::12fe:edff:fe12:60ea/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1528  Metric:1
          RX packets:3574 errors:0 dropped:115 overruns:0 frame:0
          TX packets:3512 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:144157 (140.7 KiB)  TX bytes:205127 (200.3 KiB)

bat0      Link encap:Ethernet  HWaddr 22:3f:18:58:bb:1e
          inet addr:192.168.2.72  Bcast:192.168.2.255  Mask:255.255.255.0
          inet6 addr: fe80::203f:18ff:fe58:bb1e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:42 (42.0 B)  TX bytes:468 (468.0 B)


eth0      Link encap:Ethernet  HWaddr b8:27:eb:17:7b:bb
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:4416 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7697 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:253158 (247.2 KiB)  TX bytes:11475550 (10.9 MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:194 errors:0 dropped:0 overruns:0 frame:0
          TX packets:194 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:20037 (19.5 KiB)  TX bytes:20037 (19.5 KiB)

mesh0     Link encap:Ethernet  HWaddr 22:3f:18:58:bb:1e
          inet6 addr: fe80::ccc3:3fff:fec3:ec66/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:468 (468.0 B)
```

b) **Mixing Non-B.A.T.M.A.N systems with batman-adv** (Gateway + AP Settings)

Setup your Pi as if you were setting up your computer to be an access point. Here the node has a possibility to be configured as an AP or a gateway. (This will be explained in later sections).

```
pi@raspberrypi ~ $ sudo nano /etc/network/interfaces

auto lo
iface lo inet loopback

auto eth0
iface eth0 inet manual


auto mesh0
iface mesh0 inet static
address 192.168.2.100
netmask 255.255.255.0
#gateway 192.168.2.1


#allow-hotplug wlan0
#iface wlan0 inet manual



pi@raspberrypi ~ $ sudo /etc/init.d/networking restart #Restart Networking
```

Note: The gateway setting is optional and will be used when one of the nodes is configured to be a gateway (discussed later).

Once the Networking is restarted, the `ifconfig` should read similar to the following:

```
pi@raspberrypi ~ $ ifconfig
ah0       Link encap:Ethernet  HWaddr 10:fe:ed:12:60:ea
          inet6 addr: fe80::12fe:edff:fe12:60ea/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1528  Metric:1
          RX packets:3574 errors:0 dropped:115 overruns:0 frame:0
          TX packets:3512 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:144157 (140.7 KiB)  TX bytes:205127 (200.3 KiB)

bat0      Link encap:Ethernet  HWaddr 22:3f:18:58:bb:1e
          inet6 addr: fe80::203f:18ff:fe58:bb1e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:42 (42.0 B)  TX bytes:468 (468.0 B)


eth0      Link encap:Ethernet  HWaddr b8:27:eb:17:7b:bb
```

```
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:4416 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7697 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:253158 (247.2 KiB)  TX bytes:11475550 (10.9 MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:194 errors:0 dropped:0 overruns:0 frame:0
          TX packets:194 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:20037 (19.5 KiB)  TX bytes:20037 (19.5 KiB)

mesh0     Link encap:Ethernet  HWaddr 22:3f:18:58:bb:1e
          inet addr:192.168.2.72  Bcast:192.168.2.255  Mask:255.255.255.0
          inet6 addr: fe80::ccc3:3fff:fec3:ec66/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:468 (468.0 B)
```

4. Testing the Network

To view the possible hops =>

- `sudo batctl o`

```
[B.A.T.M.A.N. adv 2013.2.0, MainIF/MAC: ah0/10:fe:ed:21:a9:95 (bat0)]
Originator   last-seen (#/255) Nexthop  [outgoingIF]: Potential nexthops
10:fe:ed:12:60:ea 0.850s(242)10:fe:ed:12:60:ea[ah0]:10:fe:ed:12:60:ea (242)
```

Layer 2 Ping =>

- `sudo batctl ping 10:fe:ed:12:60:ea`

Layer 3 Ping =>

- `sudo ping 192.168.2.72`

iperf (to test the bandwidth capabilities) =>
After installation, run `sudo iperf -s -D` on all the Batman-adv systems. TO test the performance=>

- `sudo iperf -c 192.168.2.72`

- `sudo batctl if`

```
ah0: active
eth0: active
```

- `sudo batctl tg`

```
Globally announced TT entries received via the mesh bat0
       Client         (TTVN)     Originator    (Curr TTVN)  (CRC   ) Flags
 * b8:27:eb:ed:ae:bf (41)  via 10:fe:ed:12:60:ea  (41)      (0xc4e1) [...]
 * 66:a4:82:82:8a:cd (29)  via 10:fe:ed:12:60:ea  (41)      (0xc4e1) [...]
```

- `sudo batctl tl`

```
Locally retrieved addresses (from bat0) announced via TT (TTVN: 29 CRC:
0xd643):
       Client              Flags   Last seen
 * 66:14:e8:e8:f7:9b       [.P...]   0.000
 * b8:27:eb:17:7b:bb       [.....]  86.340
```

5) VAP and Gateway Setting

Setting up access to the private mesh is a very important part of development. This can be done either by assigning one of the nodes to be a Gateway (to maintain connectivity with the Internet) or by setting up one of the nodes to be an Access Point (to enable connectivity with users / external systems).

a) Gateway Assignment

Setting up a node as a Gateway to the Internet is fairly simple since all we need is 2 active working hardware interfaces – The Wireless Adapter to maintain the Mesh + Ad-Hoc connectivity & the Cat5 Interface to connect to the IISc LAN.

Once the batman-adv mesh network is set up and the IP has been assigned to the bridge (mesh0), a second IP (of a different class) has to be assigned to eth0.

This is the IP that has to be consistent with the IP Class of the Public Network/ WAN(here IISc LAN) => 10.32.33.xx

Hence going in lines of the earlier cases =>

```
pi@raspberrypi ~ $ sudo nano /etc/network/interfaces

auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 10.32.33.221
netmask 255.255.255.0
gateway 10.32.33.221
network 10.32.33.0
broadcast 10.32.33.255

auto mesh0
iface mesh0 inet static
address 192.168.2.100
netmask 255.255.255.0
gateway 192.168.2.100


#allow-hotplug wlan0
#iface wlan0 inet manual


sudo /etc/init.d/networking restart   #Restart Networking
```

Note: The gateway setting is optional and will be used when one of the nodes is configured to be a gateway (discussed later).

Once the Networking is restarted, the ifconfig should read similar to the following:

```
pi@raspberrypi ~ $ ifconfig
ah0         Link encap:Ethernet  HWaddr 10:fe:ed:12:60:ea
            inet6 addr: fe80::12fe:edff:fe12:60ea/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1528  Metric:1
            RX packets:3574 errors:0 dropped:115 overruns:0 frame:0
            TX packets:3512 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:144157 (140.7 KiB)  TX bytes:205127 (200.3 KiB)

bat0        Link encap:Ethernet  HWaddr 22:3f:18:58:bb:1e
            inet6 addr: fe80::203f:18ff:fe58:bb1e/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:1 errors:0 dropped:0 overruns:0 frame:0
            TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:42 (42.0 B)  TX bytes:468 (468.0 B)


eth0        Link encap:Ethernet  HWaddr b8:27:eb:17:7b:bb
            inet addr:10.32.33.221  Bcast:192.168.2.255  Mask:255.255.255.0
            UP BROADCAST MULTICAST  MTU:1500  Metric:1
            RX packets:4416 errors:0 dropped:0 overruns:0 frame:0
            TX packets:7697 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:253158 (247.2 KiB)  TX bytes:11475550 (10.9 MiB)

lo          Link encap:Local Loopback
            inet addr:127.0.0.1  Mask:255.0.0.0
            inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING  MTU:65536  Metric:1
            RX packets:194 errors:0 dropped:0 overruns:0 frame:0
            TX packets:194 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:20037 (19.5 KiB)  TX bytes:20037 (19.5 KiB)

mesh0       Link encap:Ethernet  HWaddr 22:3f:18:58:bb:1e
            inet addr:192.168.2.72  Bcast:192.168.2.255  Mask:255.255.255.0
            inet6 addr: fe80::ccc3:3fff:fec3:ec66/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:0 errors:0 dropped:0 overruns:0 frame:0
            TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:0 (0.0 B)  TX bytes:468 (468.0 B)
```

After this, adding eth0 to the batctl bridge, we find that L3 ping is possible both at the private and public networks (within the Mesh & With the IISc LAN). Hence by setting dual IPs, our node now acts as a gateway to the internet thereby opening a wide spectrum of new possible applications.

b) Virtual Access Point (VAP) Assignment

To enable users to connect to the Mesh Network, 2 options are possible:-

1) Connecting via a Linux Operating System also running batman-adv.

2) Configuring one of the Pi Nodes to be an Access Point.
This can be done in 2 ways : -

a) Option 1: Adding a second radio to the Raspberry Pi and configure it to be part of the bridge. Do not assign an IP to it because it is not needed.

b) Option 2: If your wireless adapter chipset allows the VAP feature; simply create a virtual interface and configure it to be part of the bridge. For example call it mon.wlan0.

We can verify if our Chipset meets the VAP requirements by :-

```
pi@raspberrypi ~ $ sudo iw phy0 info (or ah0 info)

....

Supported interface modes:

* IBSS
* managed
* AP
* AP/VLAN
* monitor
* P2P-client
* P2P-GO

Software interface modes (can always be added):

* AP/VLAN

....
```

- As mentioned earlier, Atheros Chipsets not only have the ability to support Mesh networks by enabling the Ad Hoc mode, it can also support VAP as well. So our adapters work for the intended purpose.

- For enabling VAP, we can use hostapd or similar applications.

- Our node will now broadcast 2 SSID's. One will make the mesh among nodes on Layer 2 and the other will be used to connect users that will use Layer 3 only which we can then use wirelessly for the streaming the same way as streaming when using Cat5 cable.

The `ifconfig` looks now as follows:

```
pi@raspberrypi ~ $ ifconfig
ah0       Link encap:Ethernet  HWaddr 10:fe:ed:12:60:ea
          inet6 addr: fe80::12fe:edff:fe12:60ea/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1532  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:542 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:32012 (31.2 KiB)

bat0      Link encap:Ethernet  HWaddr 86:29:78:5a:48:87
          inet6 addr: fe80::8429:78ff:fe5a:4887/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:42 (42.0 B)  TX bytes:936 (936.0 B)

eth0      Link encap:Ethernet  HWaddr b8:27:eb:17:7b:bb
          inet addr:192.168.2.72  Bcast:192.168.2.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:164 errors:0 dropped:0 overruns:0 frame:0
          TX packets:164 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:10920 (10.6 KiB)  TX bytes:10920 (10.6 KiB)

mesh0     Link encap:Ethernet  HWaddr 86:29:78:5a:48:87
          inet6 addr: fe80::8429:78ff:fe5a:4887/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:468 (468.0 B)

mon.wlan0 Link encap:UNSPEC  HWaddr 10-FE-ED-12-60-EA-00-00-00-00-00-00-
          00-00-00-00
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2854 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:718854 (702.0 KiB)  TX bytes:0 (0.0 B)
```

So, typically we will have 3 LAN working interfaces on a gateway + AP node whether they are all physical or with some virtual.

One interface uses Cat5, another will be to form the mesh in Ad Hoc mode and the other interface in infrastructure mode to be used by wireless clients access the network.

The users will only use the Wireless Infrastructure interface and Cat5 interface.

Batman-adv will use the Ad hoc interface to forward their traffic as well as the feed and this interface will not be used by users to connect themselves.

6) Video Streaming & Port forwarding

As explained earlier, all our Layer 3 Tools can be used on top of batman-adv since it is Network Layer agnostic. IPv4 & IPv6 port forwarding can also be done without hassle.

We treat the Network the same way as we would treat a Layer 3 Network and as an addition even a wired network. Its just that batman-adv works behind the scenes to present to us a Layer 2 mesh setup.

a) Video Capture & Compression

So for our purposes of Video Capture-Compression-Streaming, we use Motion. Other alternatives as ffmpegstreamer/ gstreamer/ cheese etc.

1. Get the software ready

This needs to be done when the Pi has LAN/Internet Connectivity. Make sure the Pi has the latest drivers installed.

```
pi@raspberrypi ~ $ sudo apt-get update
pi@raspberrypi ~ $ sudo apt-get upgrade
```

Now we need to install the Motion software, this will do a few things for us including accessing the USB cam, getting the images, and streaming them via a built in web server. As the name suggests it will also track and trigger events on motion been detected in the video frames (more on that later).

```
pi@raspberrypi ~ $ sudo apt-get install motion
```

2. Plug in the web cam (to the USB Hub)

So now the software is on, it's time to plug in the web cam and ensure that everything is working.

When plugged in type the "`lsusb`" command, we should see a line there with our web cam manufacture that proves that we have the basic connectivity working.

```
pi@raspberrypi ~ $ lsusb
....
Bus 001 Device 002: ID 04ea:1142 Microsoft Corp.
....
```

3. Configure the software

```
pi@raspberrypi ~ $ sudo nano /etc/motion/motion.conf
```

In here there are a few basic changes that we need to perform:

- Daemon = OFF to ON
- webcam_localhost = ON to OFF

- We can change other settings such as frame rate too. (Increase the frame rate to the suitable level).
- Resolution from 320X240 to 640X480.
- The Port – Initially mention it as 60080 (And once tested we can change it to 8080 if needed).

4. Start the software

To ensure that the motion service will actually start as a daemon we need to change another configuration setting, so enter the following:

```
pi@raspberrypi ~ $ sudo nano /etc/default/motion
```

Then change the value "start_motion_daemon=no" to "yes"

Finally we can start the motion service to stream the web cam images

```
pi@raspberrypi ~ $ sudo service motion start
```

Then after about 30 seconds browse to the new web interface, which should be at the below URL (where 192.168.2.100 is your Raspberry PI's IP)

```
http://192.168.2.100:60080
```

5. Final Tweeks

We could change the web interface port to 80 (from the default 60080), so that we can just browse to the IP address without having to put :60080 at the end.

```
pi@raspberrypi ~ $ sudo nano /etc/motion/motion.conf
```

And then change "`webcam_port 60080`" to "`webcam_port 80`", save the file, and restart the motion service.

```
pi@raspberrypi ~ $ sudo service motion restart
```

View the feed (from any LAN node's browser) at: http://192.168.2.100

b) Port Forwarding & Viewing the Feed

With the Video Feed on the End-Node, if we are to stream it to the target, we use NetCat for packet transfer and mplayer to view the feed at the target.

So first we need to install Netcat on source and the target Pis (could be any Linux operating devices for that matter):

```
pi@raspberrypi ~ $ sudo apt-get install netcat
```

On the target machine we also need mplayer to play the video.

```
pi@raspberrypi ~ $ sudo apt-get install mplayer
```

when everything installed we run on the target machine

```
pi@raspberrypi ~ $ mkfifo a.mjpg
```

this is needed because mplayer doesn't detect correctly mjpeg streams without the .mjpg at the end.

Next on the target machine we run Netcat in listen udp mode on any port we want. I'm using port 5000 in this example

```
pi@raspberrypi ~ $ nc.traditional -lu -p 5000 > a.mjpg | mplayer -cache 32
-vo gl -demuxer lavf a.mjpg
```

on the source RPi we start netcat and send the raw contents of /dev/video0 to the target machine or the compressed data (of motion):-

```
pi@raspberrypi ~ $ sudo cat /dev/video0 | nc.traditional -u [YOUR TARGET IP
HERE] 5000
```

7) A Note on SD Cards & External USB devices

a) SD Cards

SD Cards are not much different than USB Pen/Thumb Drives and as such they also suffer data corruption with regular use and depending on their quality.

Abrupt/incorrect shutdowns will only facilitate the problem to happen faster and more times. Hence yanking out the power cable of the Pi while processes are still running should be avoided. And the same gesture, while booting up should be prevented altogether since it makes the files more prone to corruption.

The fact the Raspberry Pi uses a FAT32 partition for boot makes the filesystem even easier to corrupt.

As a suggestion; we can also lock the SD Card to not be writable or set the /etc/fstab to prevent writing which will avoid the filesystem corruption.

b) External USB Devices

Many a time, the plugging in of external USB Devices such as the Wireless Adapter / Mouse / Keyboard directly to the USB port of the Pi would result in a reboot. A person working with the Pi for a significant period of time would have stumbled upon this fault.

The reason being that the Pi operates on 700mA while the power supplied to it is via the standard 5V / 1A (Micro-USB) Adapter. Hence for its operations the Pi runs on the borderline of current. Now, direct plugging in of USB devices draws in additional current. And especially when power-hungry Wi-Fi dongles are plugged in, the current drawn goes overboard than the Pi can handle, thereby resulting in a reboot.

This can be avoided by using a multi-port USB hub which would help in power regulation.

Another effect of the same cause, is that sometimes, when the Pi is powered up, the HDMI shows no display on the monitor. And by going for a reboot (by plugging in the Wi-Fi dongle), the display comes back. The reason is the same, enough power required for display isn't reaching the HDMI port of the Pi which leads to this. The same effect can also be observed if the HDMI Cable is momentarily plugged out and then plugged in, the display however doesn't comeback.

So as a result => the correct way to plug in the external connections is as follows:-
1) The HDMI Cable
2) The Power Cable
3) The USB Hub that has the Mouse and Keyboard
4) The Adapter – when this is plugged in the system goes for a reboot and the display is now present.

If for some reason, the HDMI Display doesn't still work, the next best option is to operate it in safe mode. This can be done without having to boot up the Pi. Read the SD Card (via a Card Reader plugged to a PC). In the Media, a config.txt file can be located (/boot/config.txt).

1) hdmi_safe=1 #This would boot it on safe display mode. Should work, if it still doesn't =>
2) hdmi_group=2
3) hdmi_mode=4

And as last option. If none of the above methods work =>

4) Delete the config.txt file after backup and now boot the SD Card

Airmesh – Globally outsourcing the Mesh

Airmesh is a collection of the best open source mesh network technologies, packaged such that they are self configuring and produce a practical, network for various uses, including learning, as well as disaster and off grid communications as well as community networks.

In the local area (where all nodes are in wireless range of at least one other node) it will set up a routed private IPv6 network which may use the internet to join physically separated pockets of wireless mesh until they can link via WiFi (Bridging functionality will not be enabled over these links for scalability). It will also offer a landing page giving details of these web pages to anyone attempting to join the ad-hoc wireless network created by the system without installing the software.

The software is intended to be used on various Linux devices, but initially targeting Debian/Ubuntu based distributions and although it is still being developed, it is now completely functional given some conditions such as the availability of the correct kernel modules. Here too it is highly recommended to use Wireless Adapters with the RaLink/ Atheros Chipsets.

NOTE: Network Manager / WICD must not be running, and we must have a wired internet connection to receive updates and use internet based peering.

Technologies used

The system creates an Ad-Hoc wireless network with the SSID "airmesh". Using that it creates a virtual LAN using "Batman Advanced", with some filtering to ensure that local IPv4 networks are not bridged, and do not interfere with each other, so the internet connections of participants are not disturbed.

Once the Batman network is configured, the system will attempt to set up a routed private network using CJDNS based mesh routing. It may additionally peer with my CJDNS node which will allow access to a larger CJDNS based network as well as other airmesh nodes. This access may be restricted in future depending on the number of deployments and any abuse that occurs. We are also able to add our own CJDNS peers and our node will have access to the Hyperboria network assuming it is peered.

About CJDNS

Cjdns is a networking protocol, a system of digital rules for message exchange between computers. "Instead of letting other computers connect to you through a shared IP address which anyone can use, cjdns only lets computers talk to one other after they have verified each other cryptographically. That means there is no way anyone can be intercepting your traffic."

Cjdns implements an encrypted IPv6 network using public key cryptography for network address allocation and a distributed hash table for routing. This provides near zero-configuration networking without many of the security and robustness issues that regular IPv4 and IPv6 networks have. It is founded on the ideology that networks should be easy to set up, protocols should scale up smoothly, and security should be ubiquitous.

Cjdns is currently being used in the Seattle Meshnet nodes with an experimental network called Hyperboria.

About Hyperboria – The Alternative to Internet

Hyperboria is a global decentralized network of "nodes" running cjdns software. The goal of Hyperboria is to provide an alternative to the internet with the principles of security, scalability and decentralization at the core. Anyone can participate in the network by locating a peer that is already connected.

While Hyperboria is currently an invite-only network, a number of generous operators have extended open invitations by publicising their connection credentials. Using the public peers however is no longer advised due the centralization it creates.

The Hyperboria community encourages people wishing to join the network to meet their peers in the #cjdns IRC channel and become part of the community. To join, we install the cjdns software on a supported device/platform and locate a peer via IRC or the global map.

Sub 1 GHz WLAN Connectivity – An Overview

Introduction

The rapid developments in Internet-of-Things (IoT) and Machine-to-Machine (M2M) communication make it necessary to design communication systems operating in different wireless spectrum as an alternative to highly congested wireless access systems. In addition, the deployment of wireless smart meter devices is ramping up and it is expected that such devices will flood the market in the near future competing for the same wireless spectrum. The IEEE 802.11ah standardization task group is developing a global Wireless LAN (WLAN) standard that will allow wireless access using carrier frequencies below 1 GHz in the ISM (Industrial, Scientific, and Medical) band and will help Wi-Fi-enabled devices to get guaranteed access for short-burst data transmissions, such as meter data. In addition to exploiting the underutilized sub 1 GHz spectrum the improved coverage range allows new applications to emerge such as wide area based sensor networks, sensor backhaul systems and potential Wi-Fi off-loading functions.

The rise in license-exempt wireless technologies, such as Wi-Fi and ZigBee, indicates that Internet access and data transmission are becoming increasingly wireless. For instance, the IEEE 802.11 standard and its amendments a/b/g/n define wireless transmissions at 2.4 GHz/5 GHz in the ISM frequency band and make it simple for Internet user to build up their own WLANs, e.g., such as wireless indoor networks. The widespread deployment of IEEE 802.11-based wireless networks in indoor environments and also in urban areas has led to steep increase in mutual interference problems resulting in significant performance loss due to data packet collisions and retransmissions where Internet users suffer from low data rate or even complete network disruption. Furthermore, Wi-Fi enabled devices can be found in various mobile devices, such as tablet PCs and smart phones, thus increasing even more the competition about unlicensed wireless access in the ISM band. Smart grid applications, IoT and M2M communication will further result in saturated spectrum when the same frequencies at 2.4 GHz/5 GHz are being shared. Beside the successful deployment of IEEE 802.11 devices in the 2.4 GHz/ 5 GHz frequency band, the design of RFID (Radio Frequency IDentification) devices and other IEEE 802.15.4-based sensor networks operating in the ultra high frequency (UHF) spectrum below 1 GHz took a niche in the deployment of personal area networks. With the rapid demand for unlicensed, ubiquitous access in less-interfered frequency bands, the 900 MHz ISM bands in particular have kindled new attraction not only in the research domain but also in standardization.

Even though Wi-Fi is standardized for the 2.4 GHz/ 5 GHz ISM frequency band, there is already non-standard modified Wi-Fi equipment available that operates in the 900 MHz ISM band. As for now, various vendors take the core technology, e.g., IEEE 802.11a, and change the frequency. The demand for a standardized low-frequency WLAN comes, in part, from the smart grid community, who like lower frequencies for linking to smart meters of the larger wireless coverage and lower obstruction losses when using lower UHF spectrum for wireless data transmission. A problem, though, has been the lack of interoperability between such sub 1 GHz devices. Each vendor has its own implementation, and smart grid customers do not want to be tied to one single vendor. IEEE 802.11ah will offer a variety of advantages, such as simple to use in outdoor environments in addition to excellent propagation characteristics at low frequencies and different levels of installation scenarios (license-exempt, light licensing, professional and interference reduced). High sensitivity and link margin increase the reliability of IEEE 802.11ah. In addition, energy saving strategies will be integral part of the IEEE 802.11ah standard.

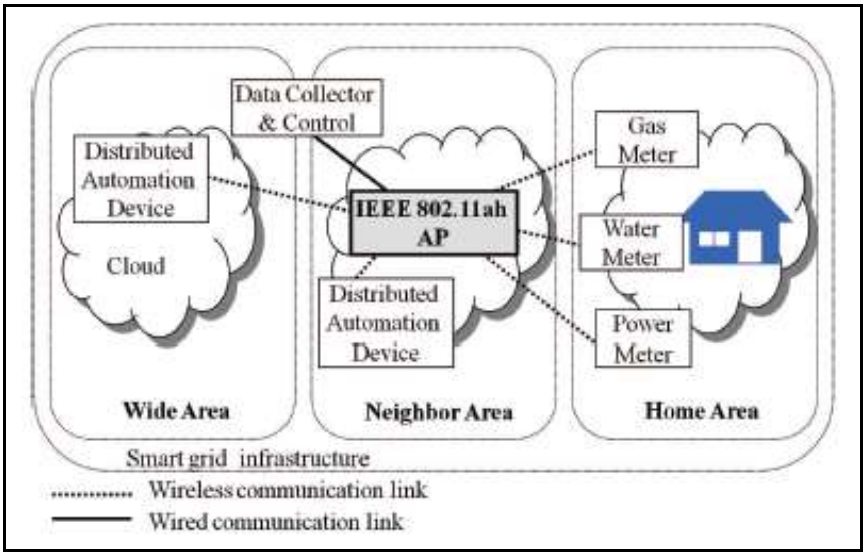

Figure: An Adopted use case of the 802.11ah – The Smart Grid

Main Advantages

In the following we outline the main advantages of a standardized sub 1 GHz WLAN:

- Longer range and less power consumed due to optimal propagation characteristics below 1 GHz.
- No licensing and regulatory issues (ISM band).
- License-exempt in various different countries.
- Almost clear co-existence issues.
- Easy to understand, follow and to implement for network device manufacturers.
- Enrichment of current wireless communication devices, e.g., IEEE 802.11a/b/g/n.

Simple Setup

- Large Coverage Area
- One Hop Reach
- Good propagation and penetration
- License-exempt, light licensing or professional installation

Reliability

- Less congested frequency band
- High sensitivity and link margin
- Diversity (frequency, time, space)

Battery Operation

- Long battery Life
- Short data transmissions
- Power saving strategies

Figure: Displaying the main advantages

Sub 1 GHz WLAN Deployment Scenarios & Use Cases

The following discussion on use cases provides comprehensive view of the advantages of using sub 1 GHz bands in various domains and scenarios.

2. *Sensor networks*

The IEEE 802.11ah includes sensor networks as one of three adopted use cases. Sensing can be executed as short-burst data transmissions and covers smart metering such as gas, water and power consumption. Wireless controlled power distribution systems are also considered. Due to the increased penetration through walls at lower frequencies, a higher number of sensors can be covered in one-hop fashion.

3. *Backhaul networks for sensors*

The second adopted use case covers the backhaul connection between sensors and/or data collectors and remote servers. The large coverage of sub 1 GHz allows a simple network design to link sub 1 GHz APs together, e.g., as wireless mesh networks. Fig. 3 shows a backhaul sensor network, including IEEE 802.11ah APs and router/gateways to connect sensor networks, e.g., IEEE 802.15.4g.

4. *Extended Wi-Fi range for cellular traffic off-loading*

The third adopted use case considers technical requirements for a Wi-Fi based cellular traffic off-loading in IEEE 802.11ah. It is important that the technology used for off-loading has at least comparable performance to the cellular system being offloaded both from the user as well as the operator perspectives. Therefore, it is essential to consider what kind of spectral efficiency, user throughput, and system load the current and future cellular networks may support.

Based on that, we need to consider the performance requirements for IEEE 802.11ah. Although it can be easily understood that a large coverage may be utilized for off-loading, some may argue that user expectations can easily be covered by existing wireless standards such as IEEE 802.11n. IEEE 802.11ah off-loading feature are expected to provide real additional value for the end user in the US market where up to 16 MHz bandwidth is available

5. *Machine-to-Machine (M2M) communication*

The future IEEE 802.11ah standard has been found as an optimal candidate as wireless communication system for M2M communication. Wireless M2M communication allows data transfer for direct machine-to-machine communication with little or no human interaction. Whereas current systems are optimized more for human-to-human (H2H) communications, IEEE 802.11ah standard will mainly consider sensing applications.

Due to different M2M standards activities happening in various standardization organizations, IEEE 802.11ah could play an important role in providing a base for a global M2M wireless standard, which some entities consider as a precursor of cloud computing. This includes smart metering, fleet management, security sensing, and on-demand business charging applications. IEEE 802.11ah will address required functions such as low power consumption, large number of devices, long-range and short-burst data transmissions.

6. *Rural Communication (connecting the unconnected)*

Wireless communication in rural areas, such as in outback areas, has lead to some effort that is also titled as *connecting the unconnected*. Proposed IEEE 802.11ah channelization for US to the wider range. E-health and e-learning would be main applications in such environments and it has been argued that a positive impact on social economics including the *Gross Domestic Product* (GDP) growth can occur.

Use Cases in Rural Environments

We present 3 use cases for rural Sub 1GHz communication systems in the following to identify the system requirements such as antenna height, bit rate, number of associated STAs and wireless coverage range.

5a. Smart Grid Networks

In the near future utility companies will start to equip their power line network infrastructure with additional sensors and meters. The goal is to make the entire utility grid *greener* by informing corporations and private end-users of their power usage, e.g., to reduce power consumption peaks which would lead otherwise to unstable operational network frequencies in the entire power line network. In rural areas a wireless coverage range up to 1km is assumed, following the proposed wireless system. Sensors, such as power, gas, or water meter will require at least 100 kbps bit rate. The proposed antenna height is between 5 and 15 meter. Meter-to-pole communication assumes that meter and sensors may be placed at the ground or even hidden behind meter boxes.

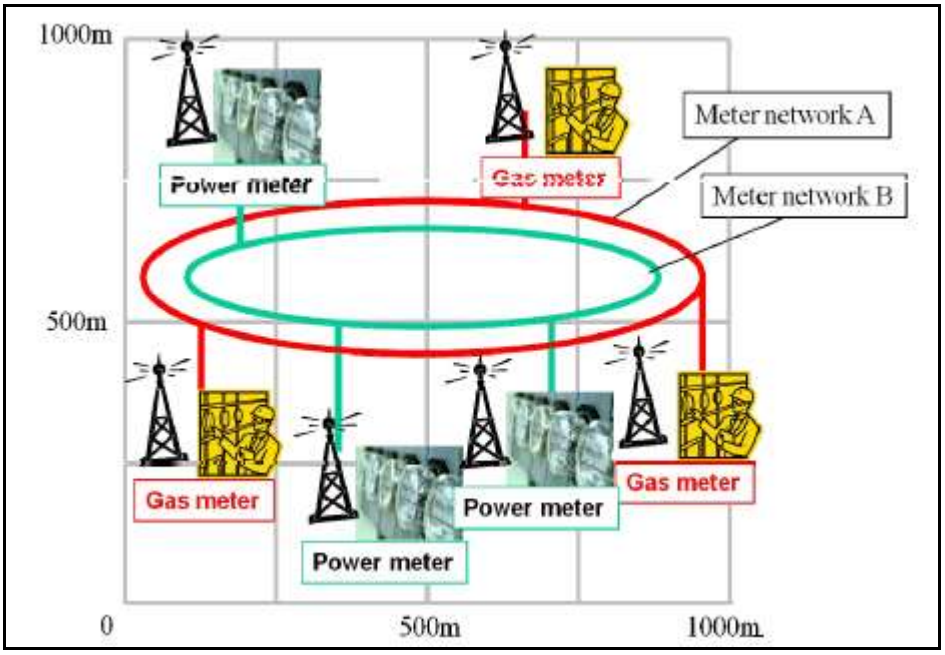

Figure: Smart Grid Networks in Rural Areas

5b. Surveillance Networks

Surveillance networks play an important role in rural areas. The monitoring of obstacles by using cameras and motion sensors are necessary to sense changes of the safety for humans and value assets. The following figure 3 shows a surveillance network that is applied in a rural area. Sensor-to-pole communication will be similar to Smart Grid, but a higher data rate such as 1000kbps is proposed, e.g., video or motion data.

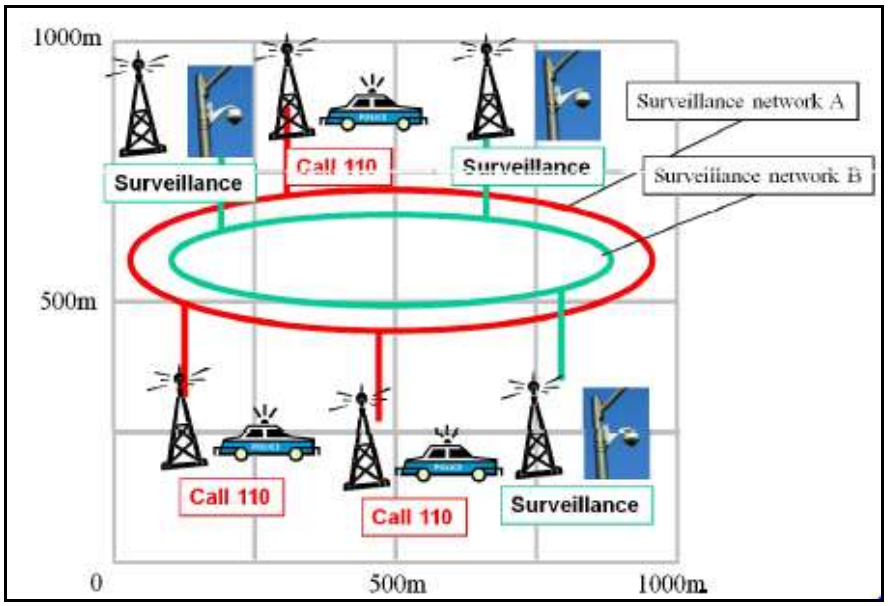

Figure: Wireless Surveillance in Rural Areas

5c. Smart Farming Networks

The deployment of sensors to detect temperature changes or outbreaks in livestock will become an important issue for farmers in the near future. Undetected flooding, fire or flu outbreaks can be devastating and entire farms can be destroyed within a short time. Figure shows a Smart Farming scenario with a sensor deployment in an outdoor area to monitor temperature gradients, water levels and livestock conditions. Data rates are at 100kbps, AP elevation is at 5m, STA antenna elevation is at 0.5m-1m.

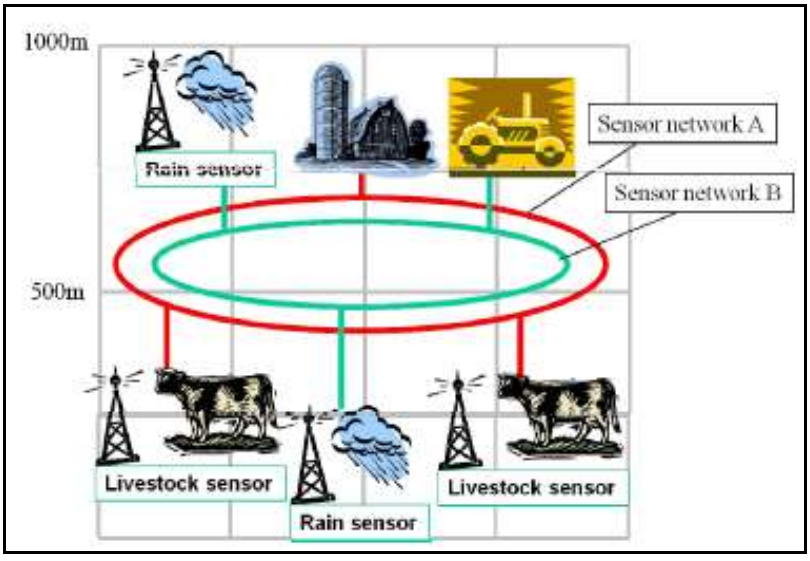

Figure: Smart Farming Networks in Rural Areas

Requirements on IEEE 802.11ah MAC

The IEEE 802.11ah MAC design considers alternative *Distributed Coordination Function* (DCF) methods. In addition, a contention-free MAC design is considered supporting a large number of stations, e.g., thousands of STAs [2], which are required for M2M and IoT applications. Power efficiency for sensor devices are proposed and are required to support the adopted sensor network use case and may include the adoption of modified IEEE 802.11v power saving features and enhanced ultra-low power consumption strategies, such as *Radio-on-Demand* (ROD).

Summary of Sub 1-GHz WLAN

With the rapid deployment of smart meter systems, IoT and M2M applications, the demand for short-burst data transmission becomes a challenging task when using current Wi- Fi frequency bands. The IEEE 802.11ah standard will define wireless access in the sub 1 GHz ISM band in various frequency domains.

Conclusion

Initially an appreciation of the state of the art technologies and applications of Ad-Hoc Networks with an exhaustive Literature Review was done. This discussed the basics of Ad-Hoc Networks as well as the current challenges and limitations faced.

Next an insight into our main objective – Wireless Video Streaming via Ad-Hoc networks was considered and the requirements imposed during development were analyzed.

Following this, an insight into the related work which involved Wireless Video Streaming through BeagleBoard-xm was done. Since this was only developed for a single hop and keeping in mind an adaptation to a new, more prevalent and better choice of development board – The Raspberry Pi was hence chosen, with a listing of its salient features and prior related applications on this emerging cutting-edge technology.

Now after the hardware listing, the next more crucial part to be focused on would be the Ad-Hoc Mesh networking and routing in particular. With an introduction to mesh networking, the package that would help us with enabling this was looked into. B.A.T.M.A.N. (Better Approach To Mobile Adhoc Networking) is a very prevalent Layer 2 technology in its current state of advanced progress that would aid us in our mission.

After looking at Zero-config with Batman-adv, a detailed description to set up the mesh network via a go-to manual was given. This gradually proceeded with initially establishing connection to the IISc LAN to the Batman-adv installation & IP Assignment. Following this was the VAP & Gateway setting thereby finally leading on to our main objective – Wireless Video Streaming & Port forwarding.

Subsequent to the Manual, outsourcing our mesh setup with Airmesh was our point of thrust. This also involved a glance into the related developments – cjdns and Hyperboria.

Coming to the final part of our progress was the Sub 1 GHz WLAN Connectivity. With a brief introduction and a listing of the main advantages, a detailed description of the deployment scenarios was studied. A special emphasis on the Use Cases in Rural Environments was given since reaching out to there is our main objective behind the Wireless Video Streaming with Mesh Networks.