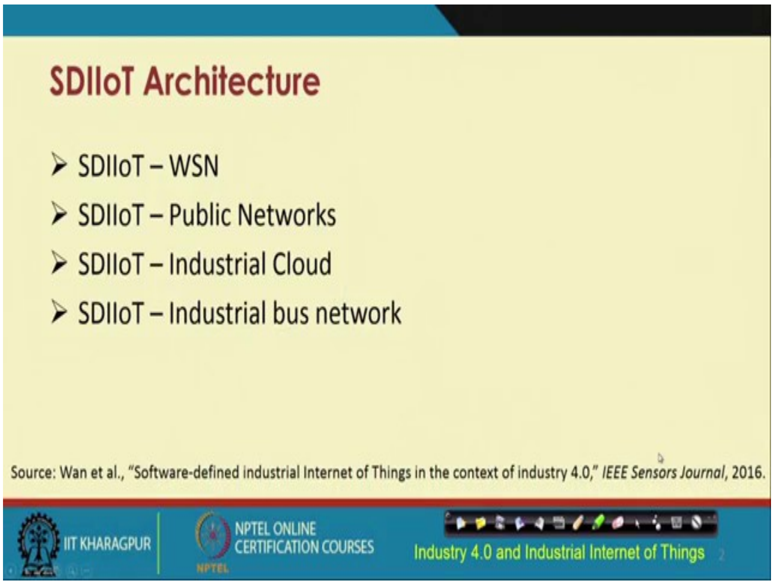
Introduction to Industry 4.0 and Industrial Internet of Things
Prof. Sudip Misra
Department of Computer Science and Engineering
Indian Institute Technology, Kharagpur

Lecture – 46
Advanced Technologies: Software – Defined Networking (SDN) in IIoT –Part 2

In the previous lecture on SDN for IIoT we looked at 2 things, first of all we understood what is the SDN architecture, what are the different components of a generic SDN architecture and thereafter we looked into this IIoT specific requirements and how SDN can integrate with a IIoT and catering to these particular requirements of IIoT in a much more efficient manner. We continue further and now we are going to look at few different solutions and applicability of SDN catering to different network scenarios in this particular lecture.

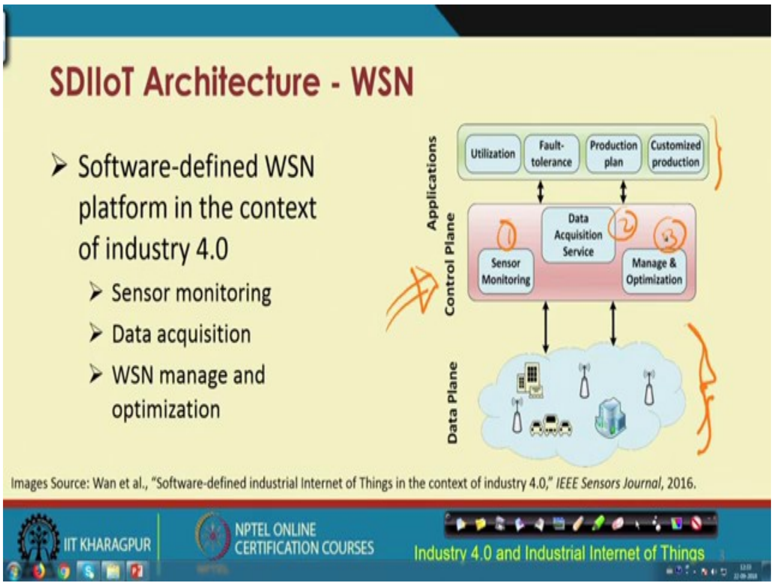
(Refer Slide Time: 01:06)



So, if we are talking about SDIIoT we have different types of networks, the traditional networks like internet public networks, sensor networks which is more specific to IIoT and you also have this cloud particularly industry grade cloud, industrial traditional bus networks connecting different sensors at the device layer and so on. So, how you are going to make them SDN enabled is what we are going to look at a very high level and particularly try to identify the main difficult areas in each of these architectures where

SDN implementation will pose challenge and how you are going to do that to cater to these specific requirements, this is what we are going to look at in this particular lecture.

(Refer Slide Time: 01:58)



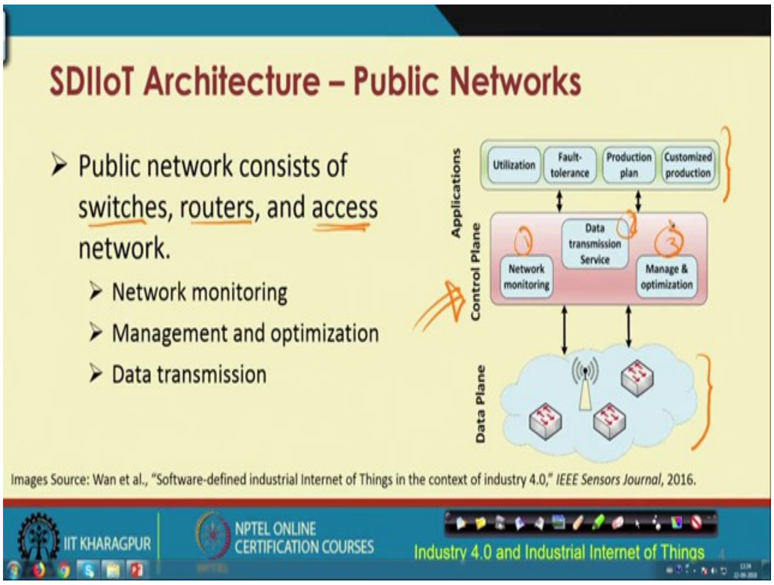
So, first let us start with the sensor network, sensors are key to IoT and IIoT. So, if we are talking about the virtualization, the software defined sensor network platforms; that means, the existing sensor networks you want to make them software defined if you want to do that what you need to take care of is basically issues of sensor monitoring, data acquisition and management and optimization of these sensors and sensor networks.

So, typically you are going to have one view of a SDN or SD enabled sensor network like this. You are going to have at the very bottom the data plane. The data plane will have all these different sensors, which may be interconnected through different access control access devices and so on. And these different devices in the data plane the sensors etc. might be there in these different transportation devices cars, buses and so on or they might be there in the different industrial, buildings or different other parts.

Then you have the control plane, this control plane basically has different components for sensor monitoring, data acquisition and management and optimization. Self optimization is very important in autonomous systems. Self optimization and self management overall has to be implemented in a software defined sensor network architecture. And on top as before we have all these different applications taking care of issues of utilization, fault tolerance, production, planning, customized production, billing, and business logic implementations, and so on so all of these different applications over here.

So, let us now focus on this particular control plane. So, we have to take care of issues of sensor monitoring, data acquisition and management and optimization particularly from an autonomous management and optimization point of view. So, these 3 components as you will notice shortly will recur in different other network settings as well, look at the internet the public networks in general.

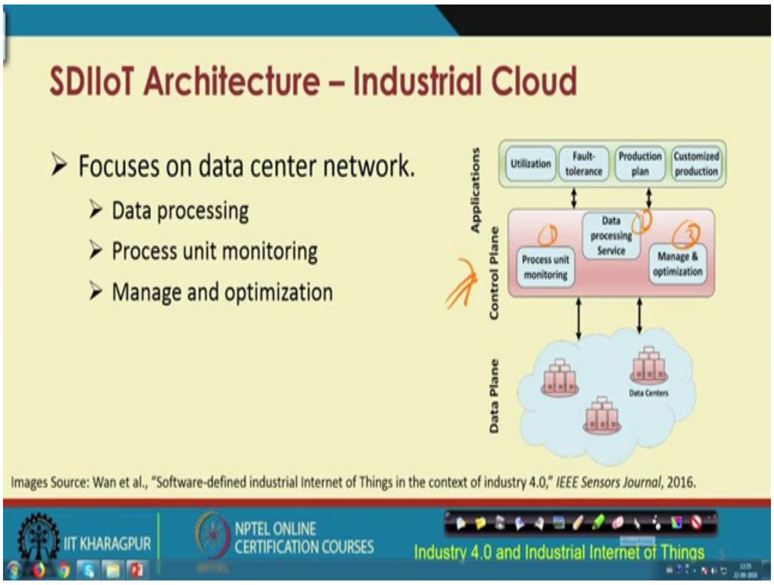
(Refer Slide Time: 04:24)



So, public networks will consist of different components such as the switches, routers and access devices. So, these are the ones that will be there in the data plane. In the application layer basically you have whatever we talked about earlier that does not change more or less, but over here in the control plane in the context of public networks you have similar kind of things like the similar kind of issues like we discussed in the context of software defined sensor networks.

So, here we are talking about network monitoring, then data transmission service and particularly autonomous management and optimization, it is very similar to the ones that you had seen in the control layer for software defined sensor networks.

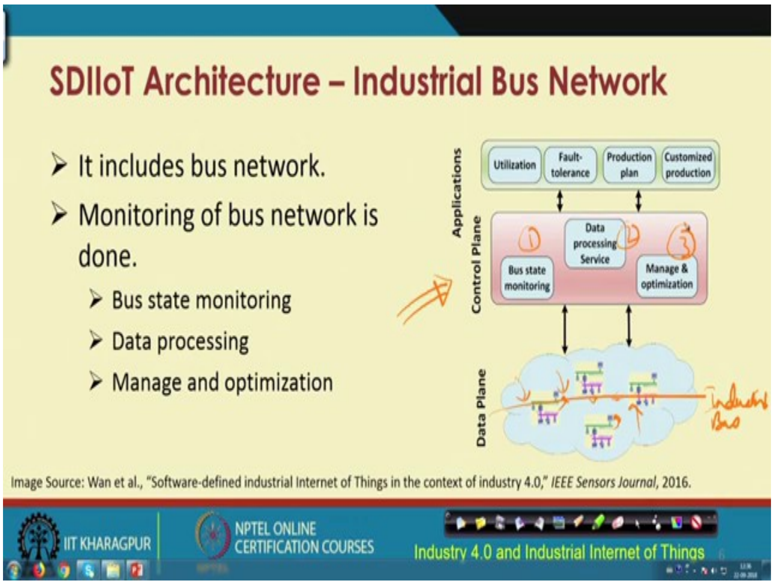
(Refer Slide Time: 05:24)



In the context of industrial cloud, cloud as we have seen is very important. Data center and cloud data center networks more specifically and cloud are very important for implementing IIoT. And software defined IIoT for industrial cloud settings you need to take care of issues like the ones over here in the control plane. So, here we have to take care of issues such as process unit monitoring, data processing service and again this management and optimization stays the same like the ones before.

So, processing and process unit monitoring these are the ones that are there in addition to management and optimization in the control plane. These are the building blocks of the control plane in the industrial cloud software defined industrial cloud in IIoT settings.

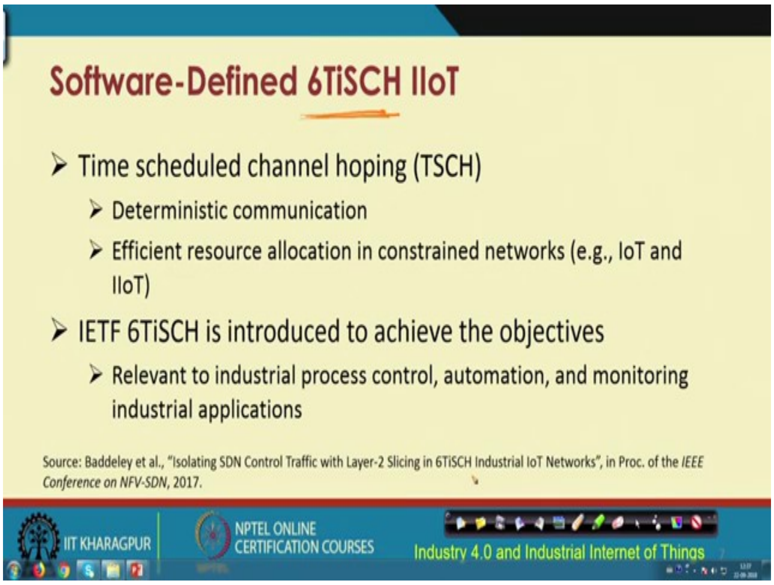
(Refer Slide Time: 06:27)



If we are talking about industrial bus network, in an industrial bus network what is going to happen? At the device layer you are going to have this industrial communication, the industrial bus, (the communication bus) to which these different sensors and other network devices are going to be fitted to the industrial bus.

So, this is your industrial bus network and to which all this different machinery with different sensors are going to be fitted and as usual on top you have these applications, but this is very important these are the different components specific to industrial bus network for the control plane in SDIIoT. So, you have over here bus state monitoring, data processing service and the management and optimization.

(Refer Slide Time: 07:21)



So, software defined networks for different settings and different architectures are all there a lot of research work is going on catering to software defined networks of all different sorts and more specifically for IoT and IIoT. IIoT industrial settings requirements are more specific, there are certain specific requirements over here.

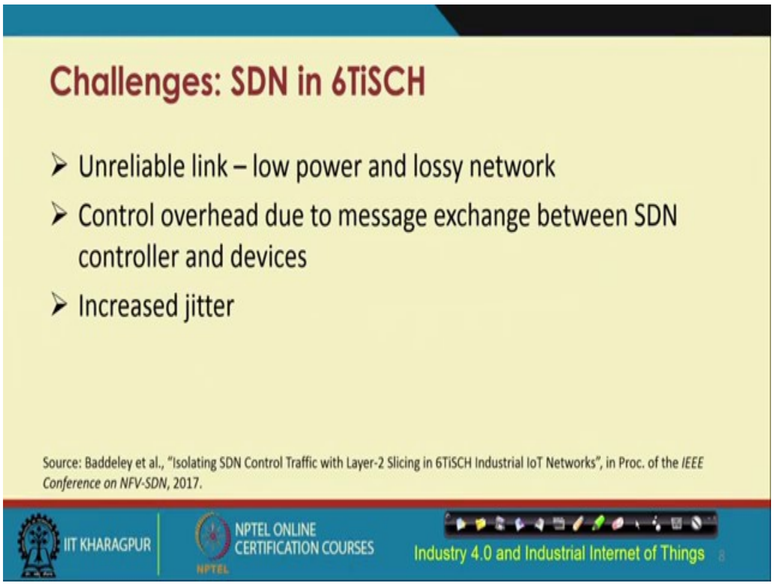
So, there is a working group which is known as the 6TiSCH working group. So, this is basically TSCH, basically stands for Time Scheduled Channel Hopping. So, time schedules channel hopping here actually what they are talking about is channel hopping in a time slotted mechanism where they are going to be time slices that are going to be there and assigning these different time slices or time slots to the different devices and the controller at the same time this is what this particular software defined 6TiSCH basically talks about.

This is a huge work that is going on in a very nutshell let me just give you the highlights, but if you need to know more beyond this, this is particular literature that you can refer to, there are so many different other literature talking about 6TiSCH particularly software defined 6TiSCH there are so many different research literature that are available for you to go through.

So, in a time schedule channel hopping TiSCH scenario we are talking about deterministic communication which is very important in industrial settings. So, this deterministic communication will help in ensuring provisioning of resource allocation efficiently in constraint networks such as IoT and IIoT because these are constraint with respect to energy, computation storage network resources and so on.

So, there is this IETF 6TiSCH working group which introduced different objectives which are relevant for industrial process control, automation, and monitoring industrial applications.

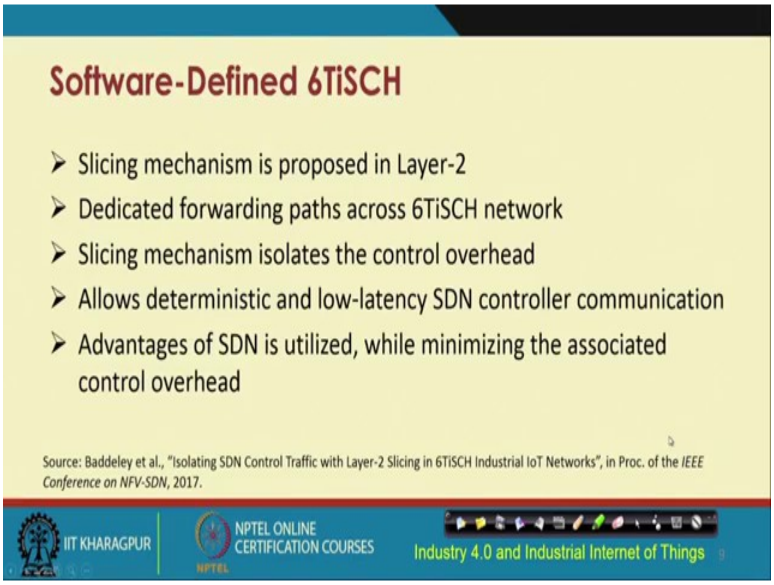
(Refer Slide Time: 09:39)



For implementation of SDN in 6TiSCH there are different challenges; challenges of dealing with unreliable links in IIoT scenarios. We have low power network scenarios, network scenarios which are lossy unreliable and so on with respect to links and components and scenarios. So, it is a highly dynamic unreliable low power highly constraint scenario where we have to implement SDN. So, this is a highly challenging job and so many research efforts are being poured in order to do so, and here is this reference that you can look at to start with in order to understand how one could think of SDN implementation in 6TiSCH.

So, control overhead is also there, because you are talking about SDN. SDN one of the important you know challenges is to deal with this overhead of control, control overhead in SDN is an important challenge. It gives lot of benefits, but it is also a challenge and how do you deal with this control overhead for this basically the slicing mechanism has been proposed.

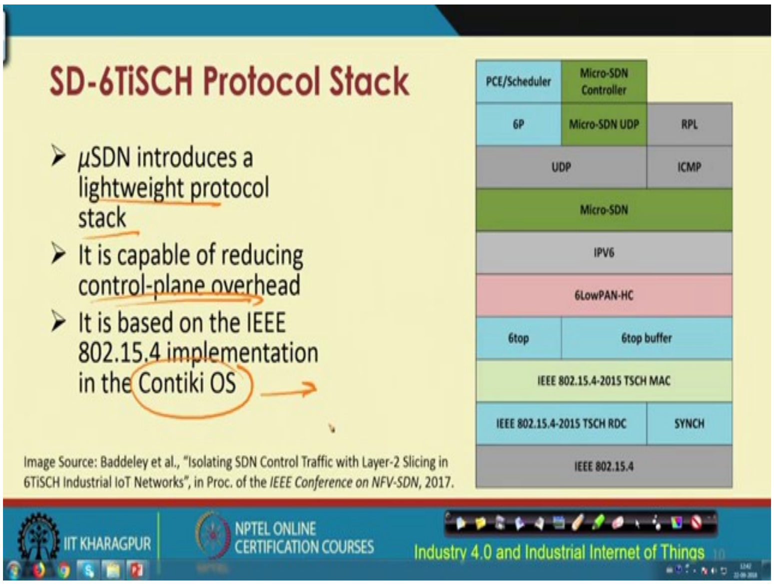
(Refer Slide Time: 10:55)



And in this particular slicing mechanism what we are talking about is to have different time slices or time slots similar kind of concepts like that and have certain devices have certain time slots the end devices, edge device you know share certain time slots the other time slots will be given to the controller. So, all of them will be using this you know the different time slices or time slots at different points of time and using them.

So, basically the slicing mechanism will give you dedicated forwarding paths across the 6TiSCH network in a much more efficient manner and will also help you in reducing the control overhead. So, basically holistically one is going to have software defined 6TiSCH providing through the slicing mechanism, providing deterministic low latency, communication for improving the performance of the network, particularly from a overall reduction of control over head and so on. So, the advantages would be that if you use SDN you are going to take care of all of these things in a much more efficient manner.

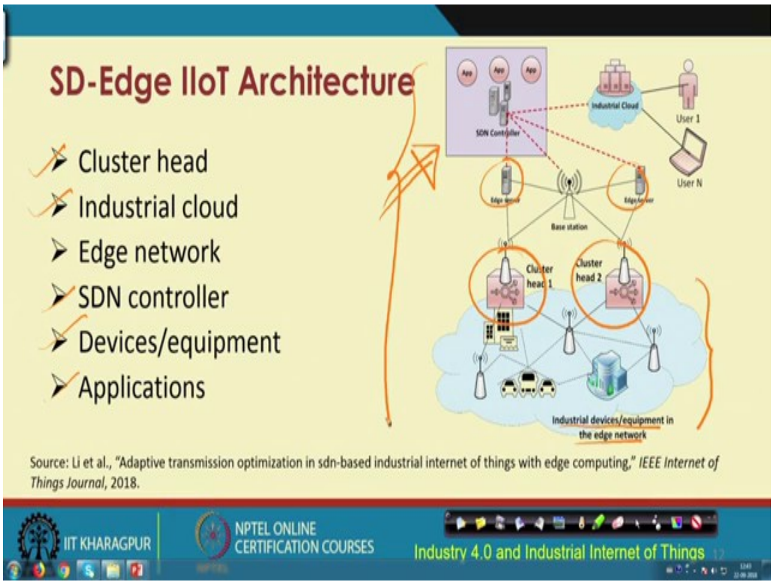
(Refer Slide Time: 12:08)



This is at very nutshell this is how this software defined 6TiSCH protocol stack looks like. So, I am not going to go through any of them in detail, but as you can see over here these are these different layers. So, layers at the very bottom 802.15.4 standard that we have talked about earlier, but these you know customized ones like the TSCH RDC, SYNCH, then you have this 802.15.4-2015 TSCH MAC and the then you have this 6 Low PAN - HC. So, 6 Low PAN basically as you know that this is a network layer protocol. 6 Low PAN we have talked about it earlier in a different lecture and this 6 basically comes from IPV6 and has been used in the 6TiSCH protocol. So, this name 6TiSCH basically the 6 comes from IPV 6 or from the 6 of the 6 Low PAN.

And then you have this concepts of the micro SDN and this micro SDN basically introduces a lightweight protocol stack that is capable of reducing the control plane overhead and it is based on the IEEE 802.15.4 implementation in this Contiki operating system which is for simulation of sensor networks Contiki is widely used. So, this basically runs on top of the Contiki operating system.

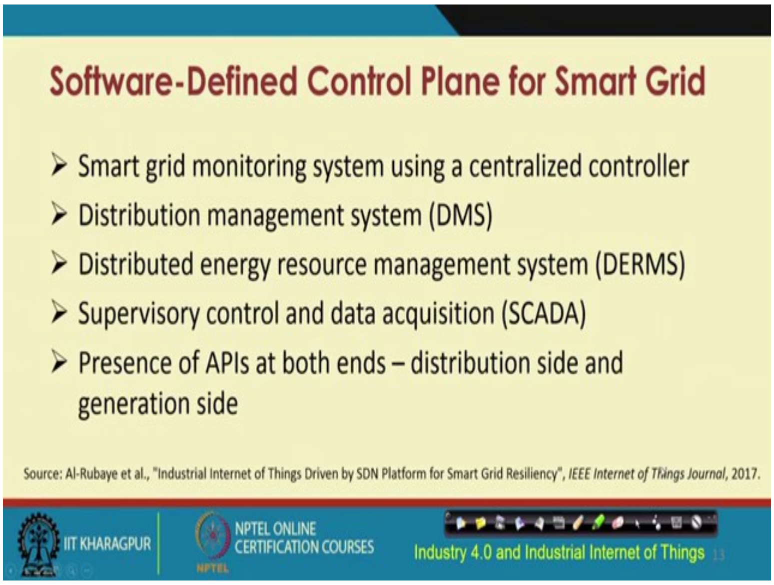
(Refer Slide Time: 13:41)



So, a software defined edge for IIoT -- this is the overall architecture you have different industrial devices and equipments in the edge network like the ones that are shown over here. And then you have these different cluster heads and you have these edge servers there after which are going to be internet work in this particular manner and then you have this SDN controller which is sitting on top in order to control the entire thing.

So, you are going to have in the software defined edge IIoT architecture different components such as the cluster head, industrial cloud, edge network, software defined network controller, devices and equipments and these applications which holistically has been shown in this particular architecture and this is quite self-explanatory. So, I do not need to go through each of these different components in further detail.

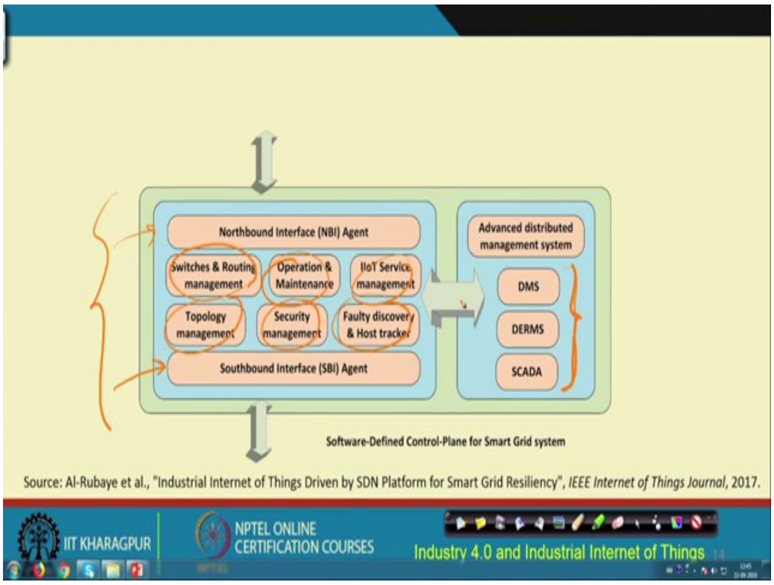
(Refer Slide Time: 14:39)



So, software defined control plane is also applicable for smart energy, smart read scenarios, in smart grid monitoring systems one could be used using the centralized controller. There are different other components in the software defined smart grid components such as the Distributed Management System the DMS, the DERMS which is basically they are Distributed Energy Resource Management System. The SCADA is there, which is important for automation as we have seen before automation and enablement of a IIoT and for SDIIoT as well SCADA is a very important component for enabling whatever we have discussed.

So, basically this particular literature in case you are interested for SDN enabled smart grid this particular literature will give you the highlights of how you are going to deal with the software defined control plane for the smart grid. So, this is the holistic view of the software defined control plane for the smart grid.

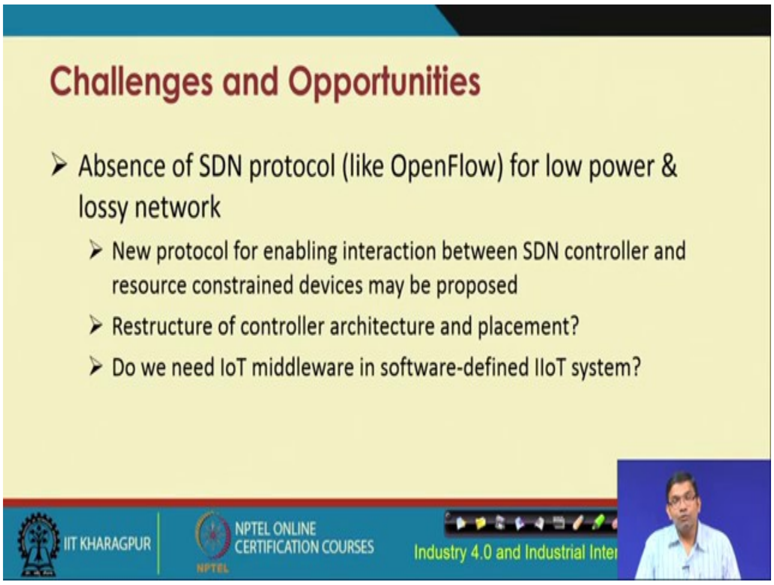
(Refer Slide Time: 15:41)



It shows only the control plane so, you are going to have all of these different components including your Southbound (SBI), North bound (NBI) and intermediate components such as switches and routers, operations and maintenance, service management, fault discovery tracking, fault tolerance in general security issues top topology management and so on. So, all of these basically are taken care of in this particular layer the control layer.

And then you have on the other side all these different components like the ones that I had shown you in the previous slide. So, SCADA, DERMS and DMS are part of this advanced distributed management system. So, together basically they work hand in hand in order to offer the software defined SDN services for smart grid.

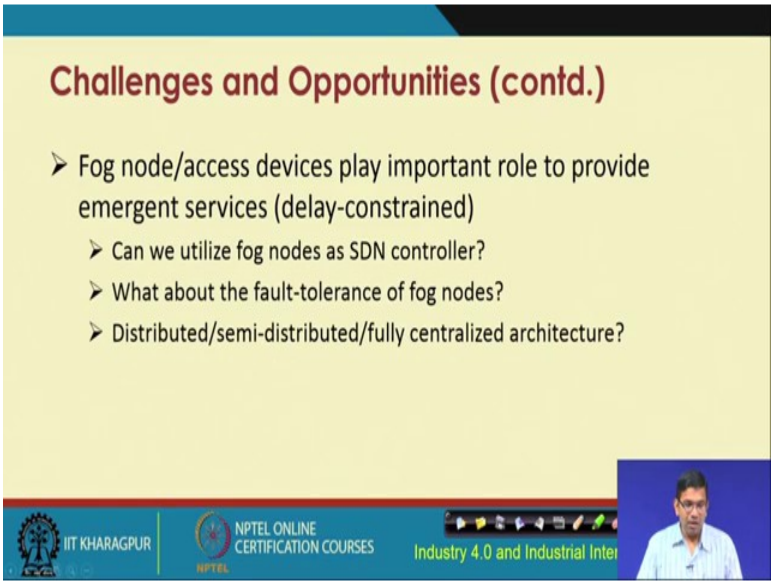
(Refer Slide Time: 16:39)



So, the challenges with respect to SDN and it is implementation in IIoT is IIoT we are talking about a highly constraint lossy network having low power and so on. SDN implementation in this kind of constant environment for example, implementation of open flow in this kind of constrained environment is required, but is a huge challenge. Open flow protocol itself is heavyweight and implementing open flow as such in IIoT constraint environments, lossy environments is a huge challenge which is quite understandable, but it has to be done as well.

So, there are consequently different works that are focusing on how you can make in open flow or other software defined solutions, light weight for implementation in these kind of constraint lossy environments of IIoT.

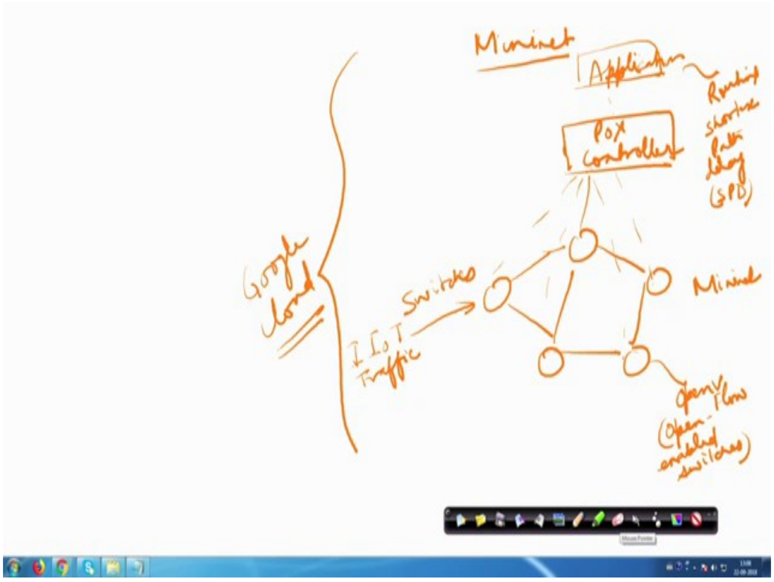
(Refer Slide Time: 17:32)



So, there are different issues different solutions for example, consideration of fog architecture where some part of the open flow or the software-defined solutions that you talked about can be implemented in the fog nodes, in the edge devices and so on and the other parts can be implemented in the centralized manner in the cloud and so on. So, there are like fog enabled solutions that are also being proposed in order to take care of these different challenges of implementing software defined networks for this constraint and lossy environments of IIoT.

We are now going to show you the implementation of open flow through Mininet which is a popular emulator that is there in the community. So, how you can use Mininet for implementing software defined networks and catering to the requirements of IIoT is what I am going to give you shortly a brief demo. So, I have with me Mr. Samaresh Bera along with me will help me in giving this particular demo. So, I am going to show you first of all how this architecture that is going to look like for implementation in Mininet. So, let me just show you this thing first that let us say that we want to implement the software defined networks in Mininet.

(Refer Slide Time: 19:04)



So, Mininet is the emulator with which we are going to emulate this software defined network scenario and we are going to use the IIoT traffic; IoT traffic more specifically. So, let us say that we have this Mininet which is going to take care of instantiation of these different nodes.

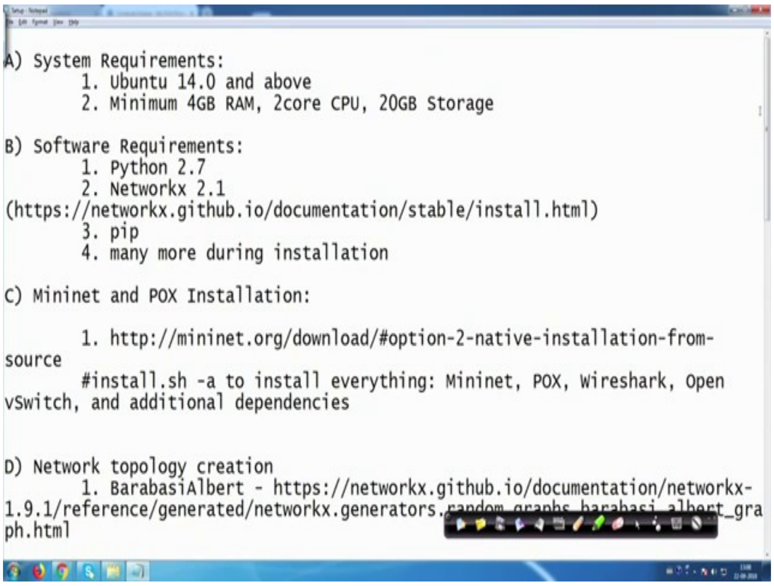
Let us say that these round circles are your different switches. So, these are your switches and that we will have a scenario like this that you are going to float some IIoT traffic through these switches and these switches will have something known as openV which is basically open flow enablement in those switches. So, we have this open rounded circles, let us assume that these are openV enabled switches.

So, which implements the open flow in it and we are going to have IIoT traffic coming through any of these different nodes and then we will have a controller. So, in SDN we have already seen that we need some kind of a controller and is the specific controller that we are going to use it is name is pox. So, pox controller is going to have this particular control over these different switches which are these openV switches right, and on top you have an application or different applications that might be running as well. So, let us say that some application is running.

So, in our case I want to show you the execution of let us see some routing protocol. The simplest routing protocol that I can think of is the shortest path delay; that means, that the shortest delay path is going to be chosen. So, in short this is known as SPD the shortest path -- the path which has the least delay is going to be chosen. So, shortest path delay protocol is going to be executed using this particular controller.

So, this is this scenario that we are going to show you now, how you are going to implement and how this routing is going to happen. And this is the Mininet environment which is executed over this Google cloud and I am going to show you how we are going to have this implementation done using Mininet emulator.

(Refer Slide Time: 22:31)

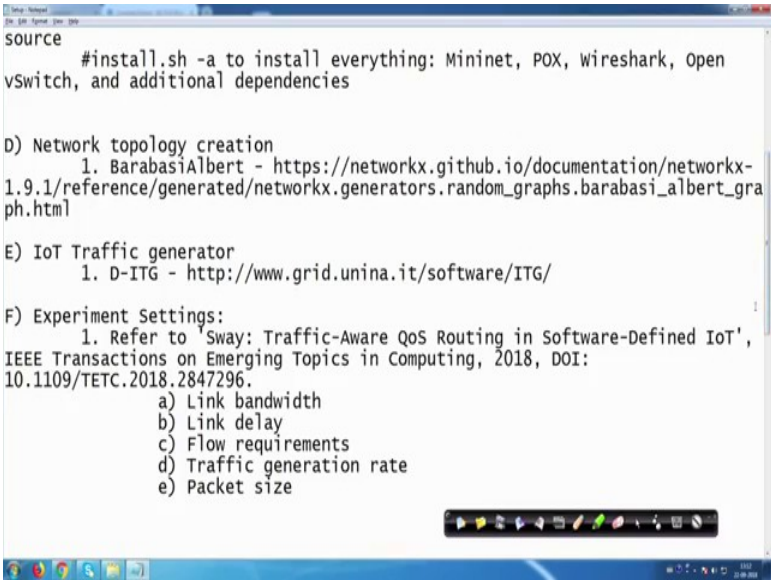


So, before I do that let me also show you something what you need to do before actually you learn this. So, for implementation first of all we need to basically have certain system settings. So, these are these different settings that will have to be done before we run our shortest path delay protocol on Mininet.

So, first of all these are the system requirements so, you need to have Ubuntu 14 and above with a minimum 4 GB RAM with 2 core CPU and 20 GB storage. The other requirements are like you know you need to install a few software python 2.7, networkx 2.1 and pip and few more installations will have to be done. Also after you have installed all of these then you have to install the Mininet and the POX. So, Mininet installation you know I have given you the source for downloading Mininet and also for installation the command that can be used is also given over here.

So, install.sh, this is going to install this mininet, pox etc. whatever the other dependencies are there so, everything is going to be installed after the downloading over here. So, thereafter we are going to have the network topology creation for that this particular package can be downloaded and it can be installed in this manner.

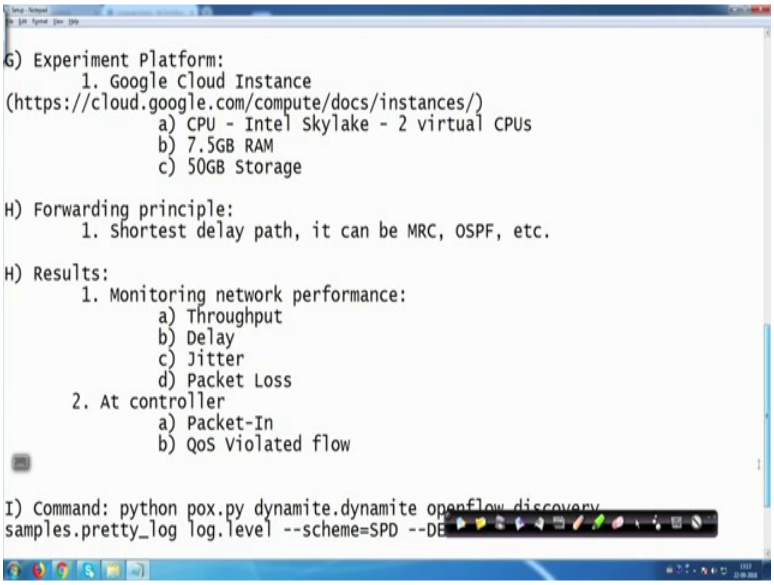
(Refer Slide Time: 24:16)



And for IoT traffic generator, this is this traffic generator that we have used; we have used the D-ITG traffic generator and it can be you know it can be procured from this particular source. And therefore, experiment settings basically you know please go through our paper which is the title of which is given over here.

It was published in the IEEE transactions on emerging topics in computing in 2018 and the corresponding DOI is also given for you. So, using this particular reference you can go through our paper the corresponding settings that are there. So, we will be using those settings for showing you this particular experiment. So, settings of the link bandwidth, link delay, flow requirements, traffic generation, rate, packet, size, etc. all of these are specified in this particular paper. So, use those settings.

(Refer Slide Time: 25:09)

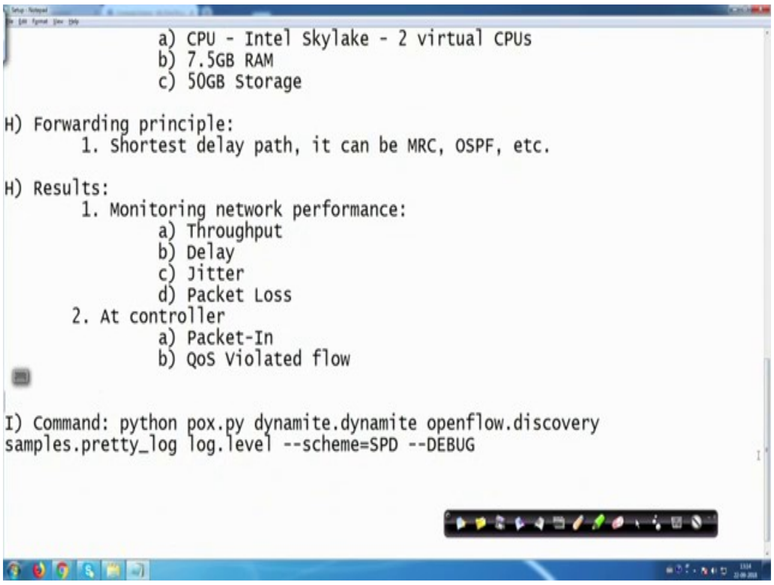


The experimental platform as I told you will be using the Google cloud and so this particular Google cloud instance we are using so, with the CPU - Intel Skylake - 2 virtual CPUs, RAM 7.5 GB and 50 GB storage and the forwarding principle will be using the shortest delay path, you could use any other routing algorithm you as well like your open shortest path first protocol or MRC or any other routing information protocol reap or whatever you want.

So, for just example sake we are going to show you the shortest delay path and you know how it is going to forward the packets from 1 point to another. So, these packets are basically routing traffic packets that we are talking about. The results that we are going to show you are basically the ones which will take care of network performance monitoring, with respect to throughput, delay, and jitter (basically the rate of change of delay with respect to time).

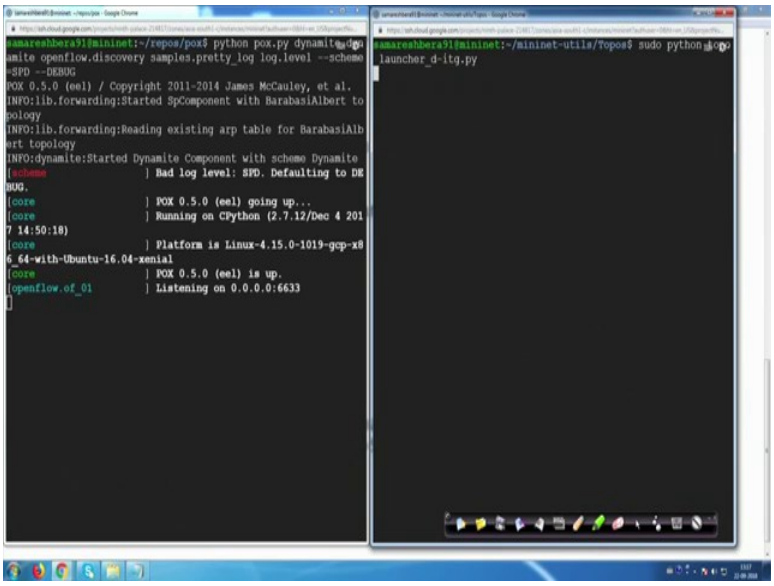
So, jitter, packet loss are some of the standard network performance monitoring parameters and these will be used, also for at the controller in the packet- in; that means, the number of packets that are coming to the controller and the QoS violations that are there. So, all of these will be measured and will be shown.

(Refer Slide Time: 26:30)



So, if you are talking about a research paper then basically you plot these network parameters, you show how these network parameters very with respect to time and. So, this is the command that you are going to use in order to execute this particular protocol that we are going to show and it is performance. So, this is this particular command that we are going to use.

(Refer Slide Time: 26:57)

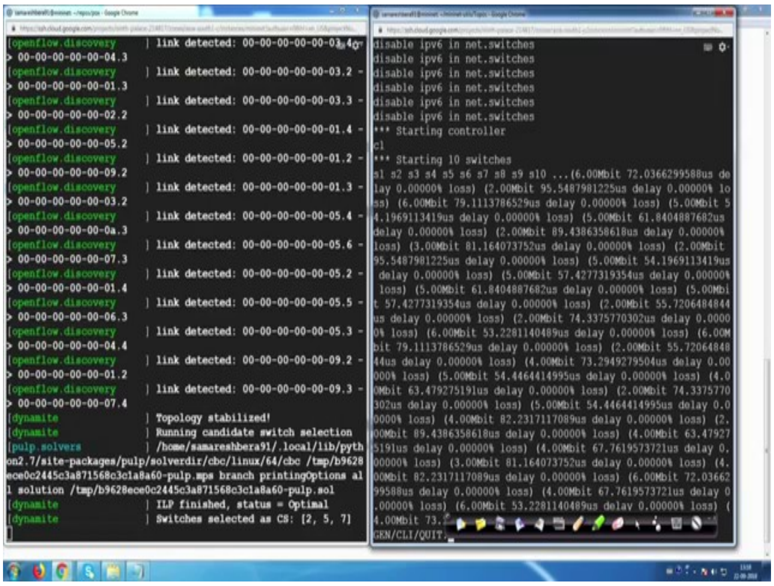


So, let us now go to this particular window and let us show you how things are going to work. So, as professor Misra mentioned that we will be using the Mininet emulator to creating the network topology using Barabasi Albert and we use the POX controller to control the switches and we are using open flow 1.1. As Mininet supports 1.1, it does not support 1.2 or 1.3. So, we will be using open flow version 1.1. So, in left side window we have the pox controller terminal and in the right side we have the Mininet topology terminal.

So, first we have to enable the POX controller so that it can listen to the switches. So, I will use some command. So, this is the command that we have written a code according to the requirements like the shortest delay path, will be running and executing the Python program, "python pox.py" which will enable all the modules of POX controller. Then we have the design scheme then open flow dot discovery so, that it can listen to all the switches whichever is been discovered. And finally, we have enabled the debugging method. So, let me run this command so, you can see the open flow is listening on port number 6633 and it is the local controller so that is why we do not have any external IP address.

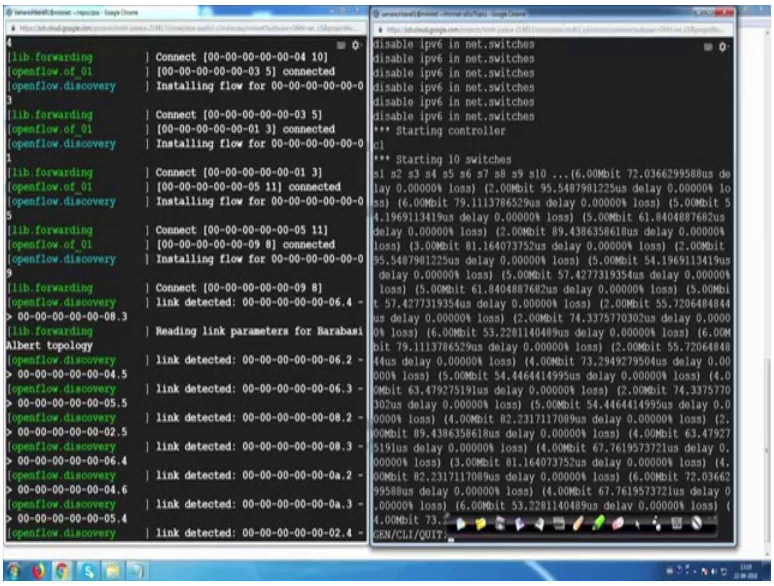
Now, the POX controller is listening. So, after enabling the pox controller we will emulate the network using Mininet. So, for that we have the specific command that "sudo python ___.py" we have created a particular script to create the topology as well as to generate the traffic.

(Refer Slide Time: 29:09)



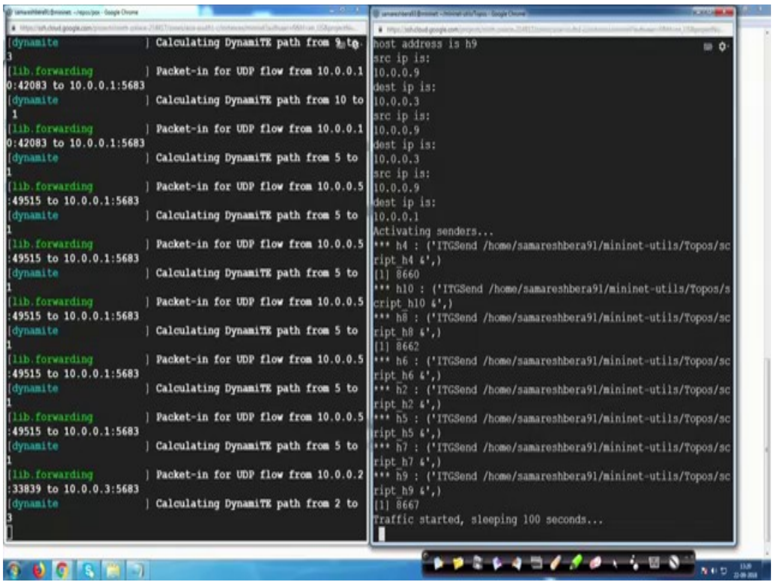
So, after emulating the network, you have created the network using Barabasi Albert topology and the pox controller listening to the switches that is why you can see that open flow dot discovery, discover different switches, the link detected, the switches detected. If I go up at the pox controller side you will see that different links are detected and these are the switches which are connected to the POX controller.

(Refer Slide Time: 29:34)



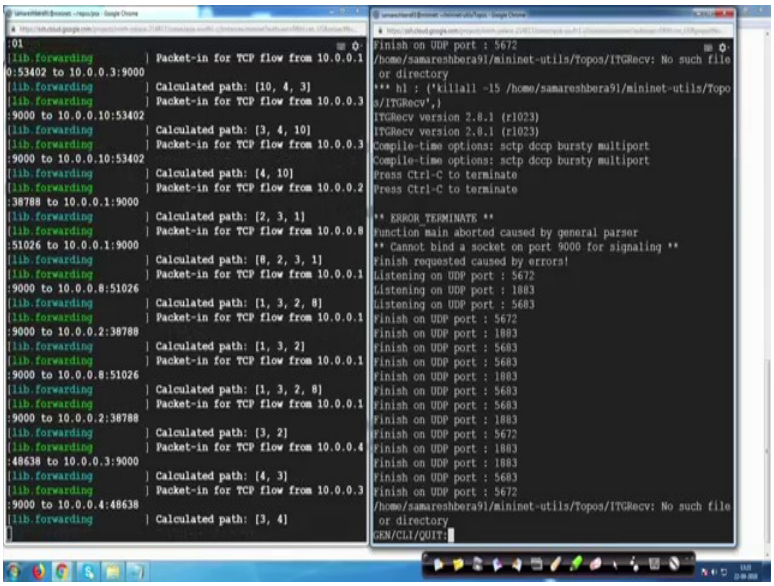
Now after creating the topology, the topology is stable and it is connected to the POX controller now using the command gen, I will generate the traffic. So, writing gen means it is enabling to generate the IoT traffic which we have defined, we have written in the script. So, if I place on gen then it is going to generate the IoT traffics.

(Refer Slide Time: 30:18)



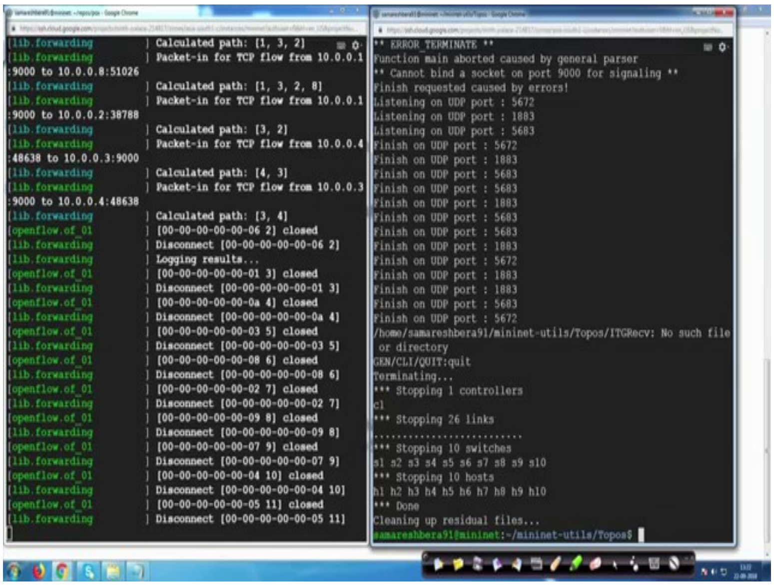
So, we have emulated the traffic for 100 seconds; that means, for 100 seconds it will generate few number of flows. So, a flow is a stream of packets; that means, we have generating few number of flows, but number of packets are in the order of 1000. So, in the left hand side at the pox controller you can see a packet in for UDP flows packet in for TCP flow. So, different packet flows are generated and the pox controller receives the packet in messages. So, just let us wait for some time to complete the experiment and then we will show you the results.

(Refer Slide Time: 31:09)



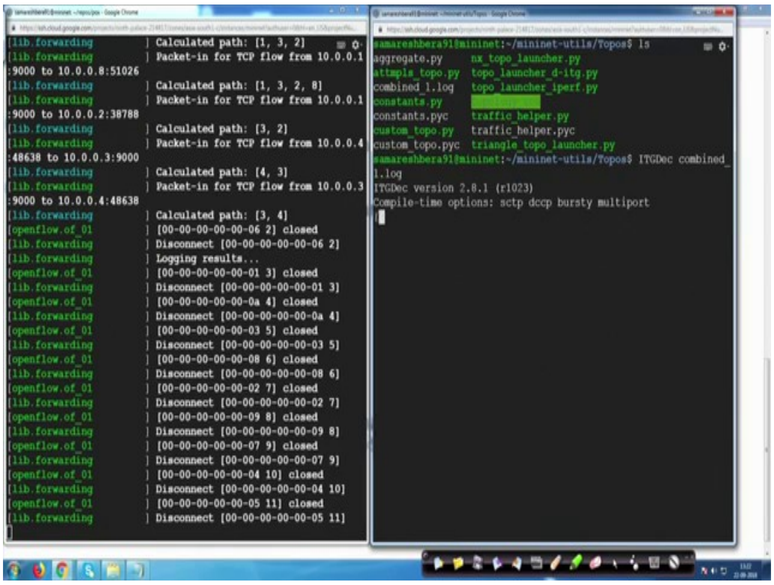
So, all the traffics are generated, the flows are generated and you can see the finish on UDP ports of different ports are generated as source. So, we have completed the traffic generation.

(Refer Slide Time: 31:23)



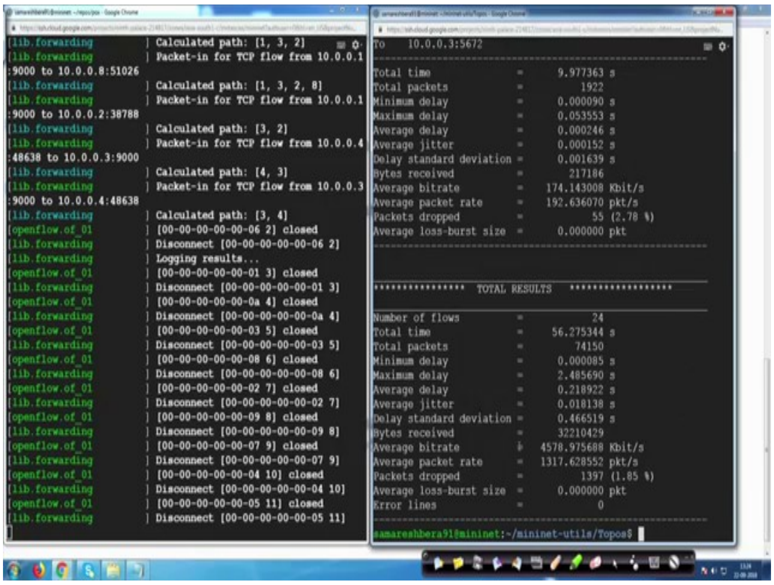
So, let us quit the Mininet emulator. So, in the left hand side you can see at the POX controller it is detected that the switches are disconnecting so, disconnected with the switch id number. Now let us show you the results.

(Refer Slide Time: 31:39)



So, here you can see at the Mininet emulator a log is generated which contains the different network performance matrix. So, let me decode it so as we have generated the traffic using D-ITG generator. So, we have to use this command ITG Decode. So, the command is “ITGDec ___.log” (lig file name).

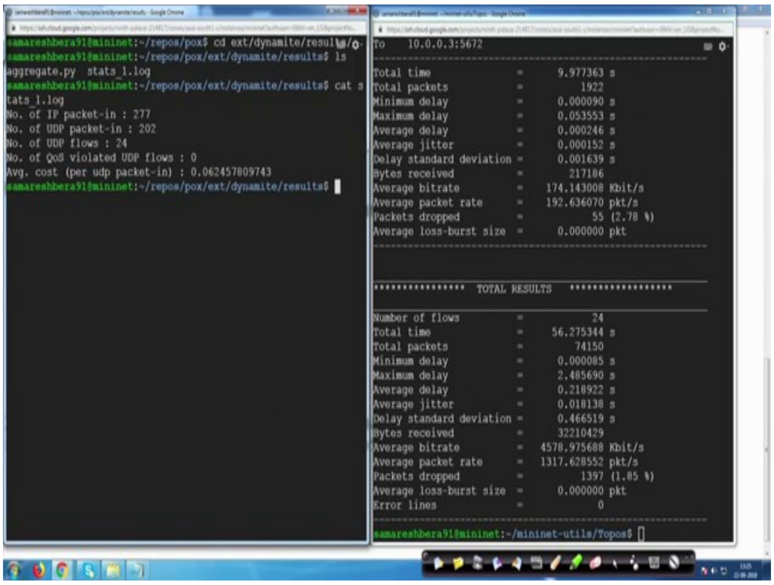
(Refer Slide Time: 32:24)



So, it is just compiling the entire thing. Now at the end you can see, we have generated the total number of flows which is 24, total time is 50 seconds. So, although we have defined 100 seconds within 50 seconds all flows are generated and routed in the network. And as I have mentioned the total number of packets is in the order of 1000s so, for 24 flows the total number of packets are generated 74,150.

And we can see different things like what is the average delay, that is 218 milliseconds, then average jitter which is 18 millisecond and then we have the average bit rate which is the throughput. So, we have got 4578 kbps and finally, at the end you can see average number of packets dropped which is 1.85 percent, which is very minimal. So, you can design your own benchmark and you can experiment it and accordingly you can measure the network performance. At the POX controller let us see what happened. I am exiting the POX controller.

(Refer Slide Time: 33:50)

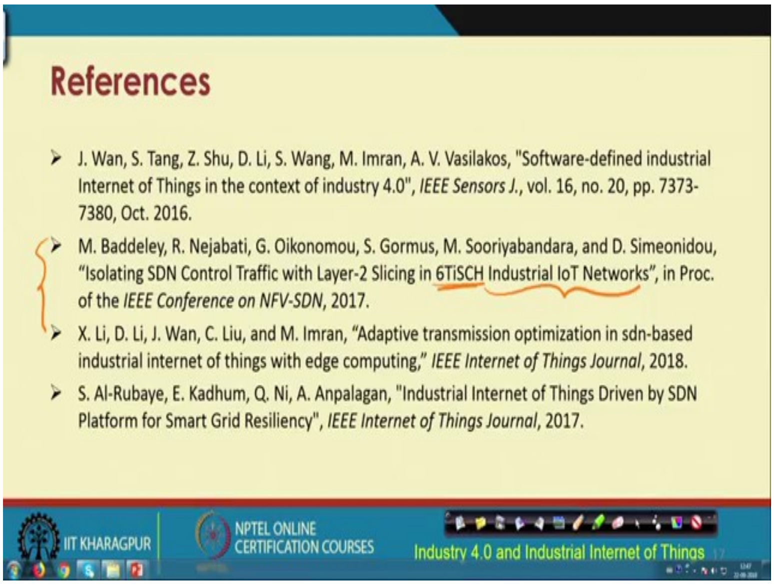


So, where we have stored the results let us see what we have obtained. So, “cat stats 1.log”, we have generated this one. So, let me check what we have got here. So, total number of IP packet we have received 277. So, number of UDP packet we have received 202 because typical as Professor Misra mentioned that typically in IoT scenario you have UDP flows. So, that is why we have counted the UDP packet also and number of UDP flows is 24.

So, in the left right hand side you can see the number of flows is also generated which is 24, number of QoS violated UDP flow which is 0. So, although we have 74,150 packets in the network, but we have got only 277 packet in messages at the controller end. That means, according to the flow rules multiple number of packets which are matched with the flow rule eventually forwarded to the destination without generating the packet at the controller rate.

So, this is a small demo we have shown to you, so that you can emulate the IoT traffic and you can a monitor the network performance, also you can phase the real data which are coming from the sensors to the network and you can deploy your own routing algorithm using the SDN controller in the real time to have, let us say, minimize delay or minimum loss or, let us say, that we want to have the maximize the network efficiency.

(Refer Slide Time: 35:47)



So, with this we come to end and this is a list of different references for you to go through further on IIoT and software defined IIoT. These references will give you a better idea about the different solutions and the different initiatives that are in place. This particular literature I would encourage you to go through in order to understand the 6TiSCH architecture and it is adoption for industrial IoT scenarios and how you could have the SDN enabled for the 6TiSCH architecture for IIoT. So, with this we come to an end of the entire lectures on software defined networks for IIoT.

Thank you.