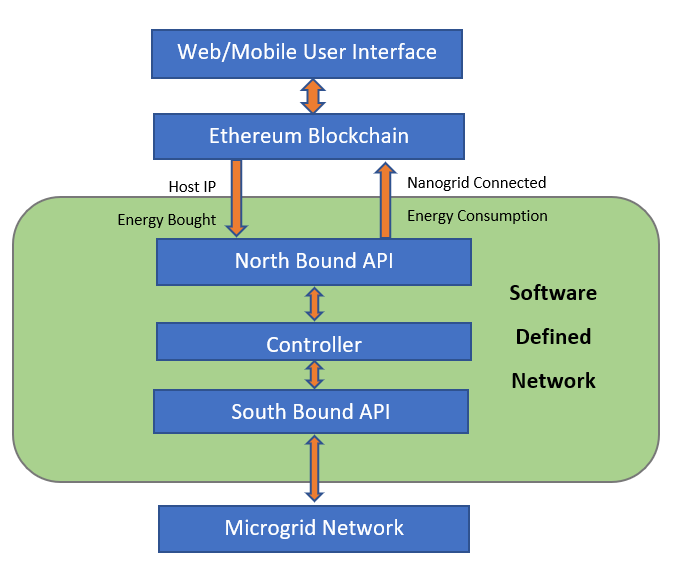
# Architectural Representation



# Architectural Components

## Software Defined Network

Software-Defined Networking (SDN) [1] is recently emerging technique that paves the way for virtualizing the network resources in an on demand manner. It provides all abstraction of the underlying network to the applications residing in upper layers. Conventionally, the network devices such as switches and routers have control plane, management plane and data plane whereas in SDN, the logic of control and data plane is decoupled separately. The control plane logic is implemented as a software component that is residing in a server and data plane is located in network devices. The decoupling of control and data plane logic has transformed the network resources into programmable, automation and network control, highly scalable and flexible networks based on the business needs. Moreover, SDN [2] replaces the functionality of networking devices as just forwarding devices. The intelligence of where and how to make forwarding is residing in control plane. The control plane logic is implemented in the software called controller. OpenFlow [3] is the protocol for communicating the controller with network devices. The SDN application 1 to N represents the features such as Quality of Service (QoS), Load Balancing (LB), Firewall (FW) and etc. that is deployed on top of SDN controllers. The Controller receives the packet and forward to OpenFlow based switches.[4]

## Northbound Application Program Interfaces

These are SDN Rest APIs used to allow communication between the SDN Controller and application running on network which might be other servers or user interface. Through our platform we connect Ethereum network to our control system.

## Control System

It acts as a mediator/controller in SDN by managing flow control to switches/router in underlying network communicating via south bound API and service/application above via north bound API.

As the central nervous system for the SDN, controller manages all the network devices in data plane, and offers the network resource calling services for the applications in the upper application plane.[5]

## Southbound APIs

These facilitate efficient control over the network and enable the [SDN Controller](https://www.sdxcentral.com/sdn/definitions/sdn-controllers/sdn-controllers-comprehensive-list/) to dynamically make changes according to real-time demands and needs. [OpenFlow](https://www.sdxcentral.com/sdn/definitions/what-is-openflow/), which was developed by the [Open Networking Foundation (ONF)](https://www.sdxcentral.com/sdn/definitions/who-is-open-networking-foundation-onf/), is the first and probably most well-known southbound interface. It is an industry standard that defines the way the SDN Controller should interact with the forwarding plane to make adjustments to the network, so it can better adapt to changing business requirements. With [OpenFlow](https://www.sdxcentral.com/cisco/datacenter/definitions/cisco-openflow/), entries can be added and removed to the internal flow-table of switches and potentially routers to make the network more responsive to real-time traffic demands.

There are a number of switch and router vendors that have announced their support of OpenFlow, including [Cisco](https://www.sdxcentral.com/cisco/datacenter/), [Juniper](https://www.sdxcentral.com/juniper/), [Big Switch Networks](https://www.sdxcentral.com/listings/big-switch-networks/), [Brocade](https://www.sdxcentral.com/lumina-networks/), [Arista](https://www.sdxcentral.com/arista/), [Extreme Networks](https://www.sdxcentral.com/listings/extreme-networks/), [IBM](https://www.sdxcentral.com/listings/ibm/), [Dell](https://www.sdxcentral.com/listings/dell/), [NoviFlow](https://www.sdxcentral.com/listings/noviflow-inc/), [HP](https://www.sdxcentral.com/hpe/networking/), [NEC](https://www.sdxcentral.com/nec/netcracker/), among others.[6]

# Process View

## Control System

* + Detection of failing pv on Panel.
  + Switch from main grid to other
  + Detection of failure on four connected neighbor and nearest provider
  + IOT device on consumer meter to detect

## Analytics on Data Stored on cloud

* + Detection of attack, fishing, DDoS, fake demand.
  + Alert detection
  + Area based consumption charts
  + Predict supplies

## Storing on blockchain network

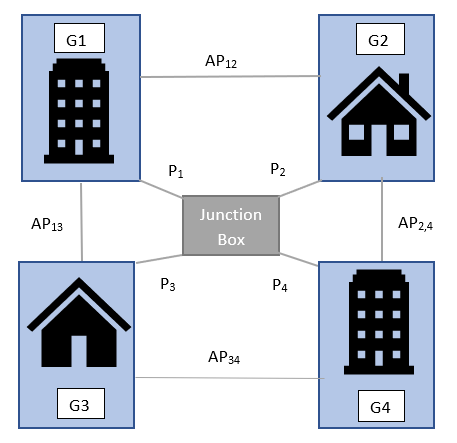
* + Store consumer and producer
  + Implement different channels
  + Only visible info
  + Fix energy units to ether
  + Only verified people

## App for producer and consumer

* + Producer and consumer interfaces
  + Info privacy
  + Buy sell screen
  + Monthly consumption statistics
  + Verified tag

# Components

## Backend / Physical Layer



As presented in the figure above we are planning to have wired links between four switches. These switches will be transferring data using primary wired interface ﬁrst connected to nanoGrid APi i [1,4] and through wired link connected to neighboring nanoGrid APj ,j[1,4], j in the 4 connected microGrid. Each of the nanoGrid is connected to a junction box through link Pi i [1,4]. Flows can be set with REST APIs of Controller. We kill a wired link, that triggers the SDN Controller, and flow continues from other nanoGrid.

When a distributed customer connects to power2peer app, these being customers with special meters to detect ip on the network essentially enabled with use of IOT device on meter, tracks the nearest microgrid and places an order. An order allows connection to the microgrid junction box enabling flow from a/some nanoGrid, part of the system, based on traffic flow. If a situation arises that a grid APi to which customer is connected currently fails, the system is resilient, and thus flow is activated from another nanogrid part of it.

As several microgrids come up, this transmission can be inter microgrid, and thereby achieving true resilience.

## Middleware / Communication Layer

### Blockchain

A SDN controller which links to single server is prone to malicious attack. With blockchain enabled solution the hosts (customers /producer/ prosumer) information is stored on blockchain network. Essential details of all three actors are verified through smart contract and regulatory authorities to identify onboarding participant on network.

* Customer
* Producer
* Prosumer

The information is stored on blockchain network as hashed identity with masked ip which is used by underlying network.

Transactions on the network and stored in non-modifiable ledger which includes

* Consumption
* Purchase

The blockchain platform thus allows a resilient power system which is safe from attacks, power theft and false hoarding.

### Monitoring and Traffic analysis

The information stored on blockchain network can be used to perform big data analytics to monitor system for power outages, study consumption graphs, regulate traffic.

## Front end

### The system provides easy mobile and web apps for ease of actors on network.

### The system provides network admin functionality UI

# Front End

## User Web/app

### Requirement

1. Register user producer/consumer
2. Producer should be able to
3. Find nearby customers/market
4. See meter reading
5. See billing record
6. See energy produced daily, yearly monthly
7. See producer Forum
8. Privacy setting
9. Allow other to see demographic details
10. Allow others to see producer posts
11. Allow other connected consumers to contact
12. Others as per regulations
13. If also acting as consumer be able to perform III.
14. Consumer should be able to
15. See nearby producers
16. See meter reading
17. Choose appliance to be connected
18. See energy consumed daily, yearly monthly
19. See consumer Forum
20. Provide rating to producer
21. Pay bill generated at the end of month(limit should be added)
22. Stop energy purchase
23. Able to buy token
24. Privacy setting
25. Allow other to see demographic details
26. Allow others to see consumer posts
27. Others as per regulations

### Functional Solution

### Technical Solution

## Administrator

### Requirement

#### Network Administrator

1. Should be able to add, remove producers, consumers host.
2. Should be able to add switches to host/switches
3. Should be able to create paths via junction to hosts to other microgrids, hosts in a microgrid
4. View traffic
5. View failed nodes status

#### User Administrator

1. Verify producers after right information is legally obtained
2. Verify consumers
3. Monitor requests for new solar producers if need solar panels installed.

### Functional Solution

### Technical Solution

# Backend

## Controller

### Requirement

1. Receive input from ethereum network containing hosts details
2. Send “host connected to” response to ethereum
3. If connection changes between host to host
4. Send updated host to ethereum network
5. Send meter reading of previous host
6. Send network info to Data Lake explained in 2.2.2.3 IV
7. Apply cap on flows
8. Specify intents for data flow between hosts
9. Detect failure

### Functional Solution

* + 1. RQ2.1.1 VII

Minimum-cost flow algorithm

Network traffic is created based on max cut off set.

* + 1. RQ2.1.1 VIII, RQ2.1.1 IX

System should be trained for all consumers to see energy consumed pattern and create network traffic accordingly, this ensures that energy supplies meets demands.

### Technical Solution

## North Bound API

### Requirement

#### Monitoring

#### Security

1. Cyber vulnerability
2. Data privacy and security

#### Data Analytics

1. A datalake needs to be built for analytics of this noSQL information.
2. The data lake will have following data buckets for real time analysis and batch processing through Datawarehouse.
   1. Meter Reading
   2. Producer consumer network
   3. Network traffic
   4. Billing records
   5. Power outages and natural calamity etc for determining supply to plan ahead
   6. etc

### Functional Solution

1. RQ2.2.1.3 I
   * + 1. Apache Kafka acts as the high-performance data ingestion layer dealing with massive amounts of data sets. IoT devices comprise of a variety of sensors capable of generating multiple data points, which are collected at a high frequency. Since these devices may not be powerful enough to run the full TCP networking stack, they use protocols like [Z-Wave](http://www.z-wave.com/), and [ZigBee](http://www.zigbee.org/) to send the data to a central gateway that is capable of aggregating the data points and ingesting them into the system. The gateway pushes the data set to an Apache Kafka cluster, where the data takes multiple paths. Data points that need to be monitored in real-time go through the hot path.
       2. These data points may go through an Apache Storm and Apache Spark cluster for near real-time processing.
       3. Metrics such as load and power consumption are analyzed after collecting them over a period. These data points that are collected and analyzed through a batch process typically take the cold path of the data processing pipeline. A [MapReduce](https://www-01.ibm.com/software/data/infosphere/hadoop/mapreduce/) job may be run within a Hadoop cluster for analyzing the energy efficiency

### Technical Solution

## South Bound API

### Virtual test network

A virtual test network can be created using mininet.

### Topology

1. The network topology consists of **junction switches**, i.e. switches connected to junction host.
2. Each provider/consumer is connected via switch to enable disable connection to the producer /consumer host called **host switch**.
3. Each host switch is connected through a switch to junction switch called **connection switch**.
4. Each producer’s host switch is connected to other neighboring producer’s host switch through **producer connection switch** in a microgrid.
5. Each microgrid junction switch is connected to other microgrid junction switch via **microgrid switch**.
6. Gateway, IP, mac address, device id, gateway, IP protocol needs to be obtained and stored securely.

|  |  |
| --- | --- |
| **Symbols** | **Representation** |
|  | Junction switch |
|  | Producer connection switch |
|  | Connection switch |
|  | Host switch |
|  | Microgrid switch |
|  | consumer |
|  | producer |
|  | Junction box |
|  | connection |

### Hardware connection

### Requirement

1. Connecting to IoT devices, microinverters on solar panels
2. Connection to solar array through microinverters
3. Managing IoT devices

### Functional Solution

* 1. RQ2.3.3.1 III

Kakfa with MQTT

In an IoT solution, the devices can be classified into sensors and actuators. Sensors generate data points while actuators are mechanical components that may be controlled through commands. For example, the ambient lighting in a room may be used to adjust the brightness of an LED bulb. In this scenario, the light sensor needs to talk to the LED, which is an example of M2M communication. MQTT is the protocol optimized for sensor networks and M2M. Since Kafka doesn’t use HTTP for ingestion, it delivers better performance and scale.

Since MQTT [is designed for low-power devices](https://thenewstack.io/off-shelf-hacker-lightweight-inter-device-messaging-mqtt/), it cannot handle the ingestion of massive datasets. On the other hand, Apache Kafka may deal with high-velocity data ingestion but not with M2M.

Scalable IoT solutions use MQTT as an explicit device communication while relying on Apache Kafka for ingesting sensor data. It is also possible to [bridge](https://github.com/evokly/kafka-connect-mqtt) Kafka and MQTT for ingestion and M2M. But it is recommended to keep them separate by configuring the devices or gateways as Kafka producers while still participating in the M2M network managed by an MQTT broker.

### Technical Solution

## Ethereum Layer

### Requirement

1. Asset
   1. Host ip
   2. Host mac
   3. role
   4. Energy purchased (Stored at end of a purchase cycle or provider change)
   5. Energy provider ( Stored at end of a purchase cycle or provider change)
   6. Bill( Escrow funds need to be maintained)
2. All other information for producers and consumer to be stored in db.
3. When a producer/consumer registers their network information is stored.
4. When a producer registers it should be verified and then added to network. Producer addition uptime will be compromised. Energy production capacity need to be provided for maintaining traffic flow.
5. When a consumer registers, smart meters should be enabled(if not already option provided) Consumer addition uptime will be compromised.
6. When a consumer trades energy a maximum supply cut off should be set on consumer and amount is deducted from consumer as stored as Escrow fund. No payment is made to producer. The mechanism should be similar to car booking, a hold is made on the account for ride fee. This is essential as it ensure producers are paid at the end.
7. At the end of billing cycle all producer and energy provided by them are stored for a consumer. Based on this bill are generated. If producer p1 generated only 4 KW and p2 generated 6KW which was provided to consumer x. P1 and P2 are paid as per power consumed based on simple rate calculation formula. The escrow funds are paid and consumer amount is deducted of price for energy consumed and not the cap.
8. If energy consumed is more than cap alert should be sent to consumer before cut off.
9. Ethereum acts a middle layer between front end and controller.
10. Ethereum network receives following information from controller
    1. Energy provider
    2. Smart meter reading
11. Ethereum network receives following information from frontend server
    1. network information for producer consumer
    2. if producer or consumer
12. Ethereum with IoT provides security essential for IoT devices.

### Functional Solution

### Technical Solution

## DB Connection

### Specifications

### Requirement

1. All user and billing information needs to be stored in db.
2. The information for a user is obtained through hashed identity stored on Ethereum network.
3. App fetches data from db to find nearest neighbors producers or consumers.
4. Sends info to analytics engine

### Functional Solution

### Technical Solution

# References:

1. Open Networking Foundation (ONF) (2013). [Online]. Available: "Software-Defined networking: the new norm for networks," <https://www.opennetworking.Org/images/stories/downloads/openflow/wpsdn-newnorm.pdf>.
2. OpenFlow Consortium (2013). [Online]. Available: <http://openflowswitch.org>
3. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69-74, 2008.

# [Kannan Govindarajan](https://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22Authors%22:.QT.Kannan%20Govindarajan.QT.&newsearch=true); [Kong Chee Meng](https://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22Authors%22:.QT.Kong%20Chee%20Meng.QT.&newsearch=true); [Hong Ong](https://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22Authors%22:.QT.Hong%20Ong.QT.&newsearch=true), “A literature review on Software-Defined Networking (SDN) research topics, challenges and solutions”, Published in: [2013 Fifth International Conference on Advanced Computing (ICoAC)](https://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6915272) , DOI [10.1109/ICoAC.2013.6921966](https://doi.org/10.1109/ICoAC.2013.6921966)

1. SDX Central[Online]. Available: <https://www.sdxcentral.com/sdn/definitions/sdn-controllers/> <https://www.sdxcentral.com/sdn/definitions/southbound-interface-api/>
2. Feng Wang, Heyu Wang, Baohua Lei, Wenting Ma, ”A Research on Carrier-grade SDN Controller”, 2014 International Conference on Cloud Computing and Big Data, DOI 10.1109/CCBD.2014.41