Objectives

Develop an extensible framework, to develop new IoT class of applications.

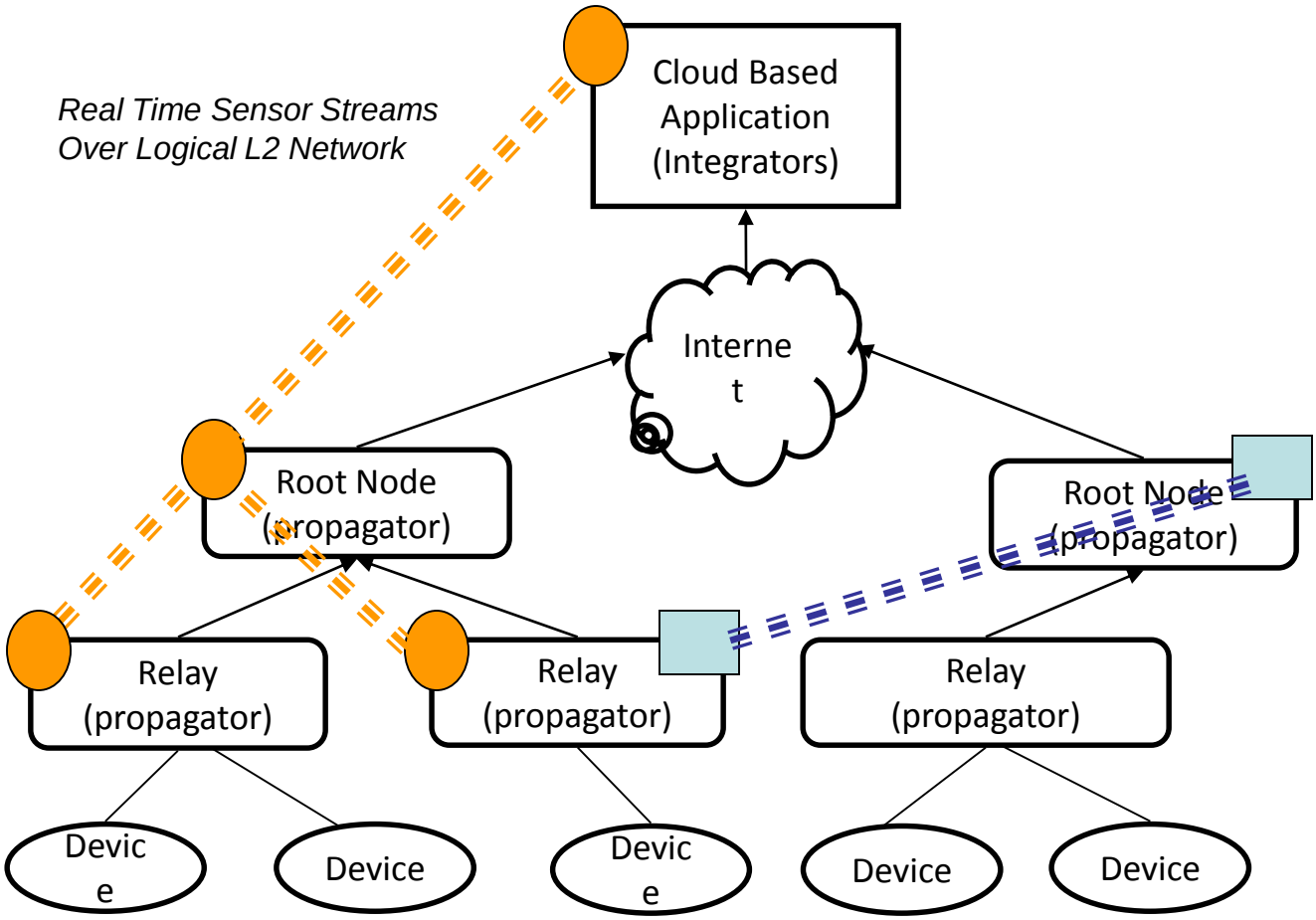
Applications should be agnostic of the physical medium, connectivity etc.

Platform to be agnostic of the application logic (tested in simulator).

Applications employ vendor neutral M2M messaging (Real Time Pub/Sub)

Architectural Components

1. Drivers
2. Protocol translation layer
3. Applications
4. Packet Scheduler
5. Agent on the Device
6. Device Registration Mechanism
7. Application Pub/Sub
8. Application Virtual Network
9. Flow based packet processing.
10. Cloud Application
11. Rules definition for the cloud application
12. Rules propagation from Cloud to Mesh Nodes
13. Communication of devices across different networks
14. Local decisions on the mesh node



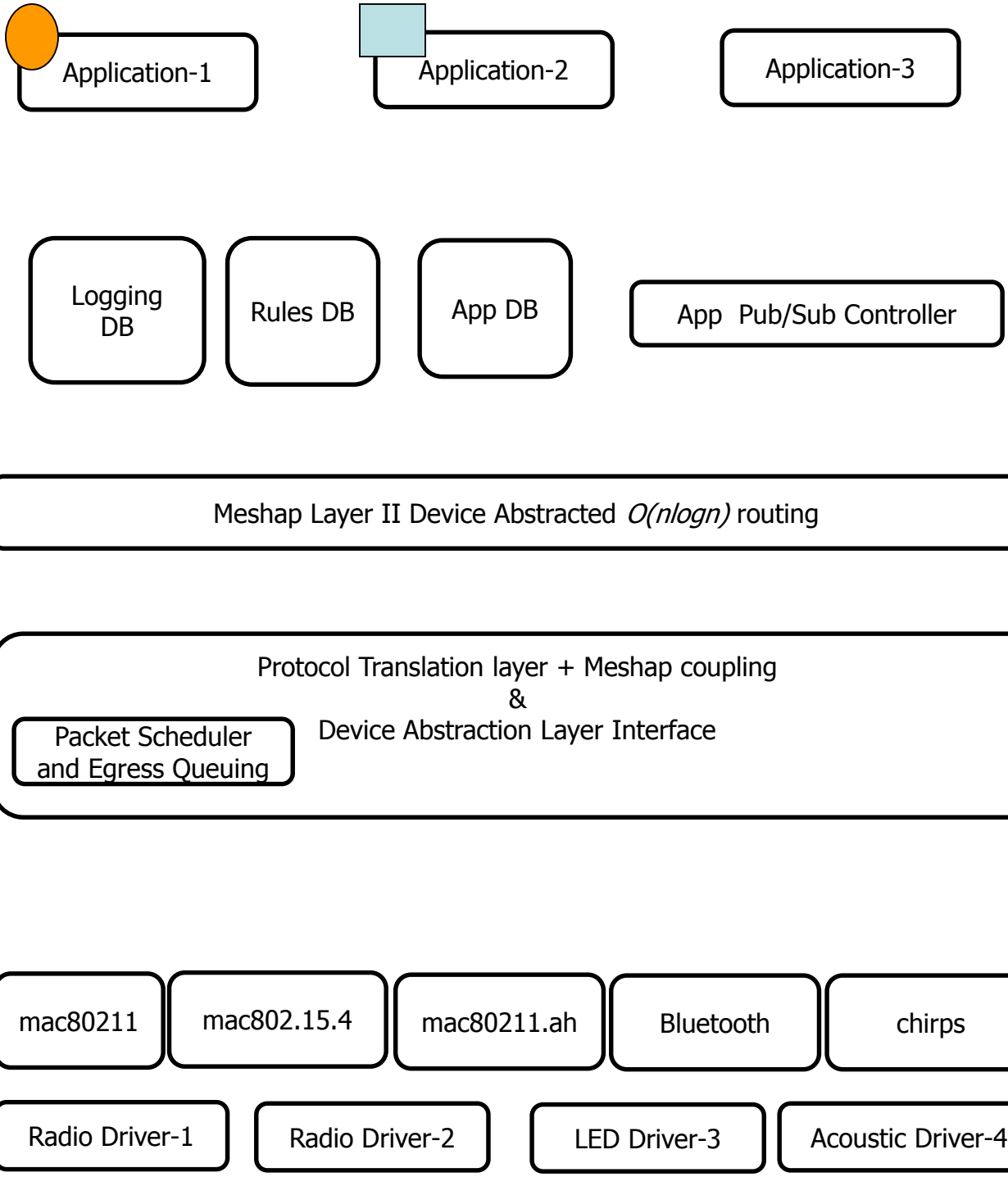
Mesh Nodes operate in multiple, disparate and possibly remote Enterprise Networks. In silo based cases, only related or "known" applications talk to each other.

The power of this platform is applications -- not related to each other – can subscribe to events from other App, discover new patterns and learn (Collaborative, not silo based)

We define the open, identification domain for the Apps and encode it in their message Apps can be decide if they wish to subscribe: Topic Based Real Time Publish Subscribe.

Example: Enterprise Apps may subscribe public events local to them but not in their Silos.

Architectural Components



Application Ids are identified as the following:

Class-of-Application/ App-Type /Location / App-Id

Class of Application, broadly identifies the various classes of application. It can be a public class or a private class specific to the application.

Public class, Application is willing to provide its identity according to a public, topic based addressing scheme. Benefit: Other applications "discover" them and subscribe: Collaborative Intelligence across large enterprise networks. .

Private class Application will talk with its sub tree in a topic based addressing scheme that is maintained by the Enterprise (Silo model).

Unique MAC_ID: To keep the values of the Application ID Instance unique across networks, Applications must register the IDs with the cloud application as part of their INIT process and only if the class is registered by the cloud or it already exists, application can proceed.

Example:- Class of Application can be *pub.lighting, priv.lighting_phillips*

App Type identifies the type of application and is unique within the class of applications

There can multiple applications running with the same class and app-type.

Example can be a led_dimmer. The application can look like:

pub.lighting/led_dimmer
priv.lighting_phillips/led_dimmer

App-ID is the unique MAC Address of each instance of the application. This identifies the end point of the app, needed for Layer II routing. This will be as part of the application sign-on and branding process. The complete APP-ID is defined as

pub.lighting/led_dimmer/india.bangalore.560048/aa:bb:cc:dd:ee:ff
priv.lighting/led_dimmer/ijapan.tokyo.1000401/bb:cc:dd:ee:ff:gg

Publish

When an APP publishes the events, it will publish the following

APPID/Event-type/Event-data

Event type and data is specific to the APPs.

Each APP Class + APP-Type must register their set of events with the cloud application.

From the above examples, the event type can look like

pub.lighting/led_dimmer/iN.BLR.560048/aa:bb:cc:dd:ee:ff/power-level
priv.lighting/led_dimmer/US/SFO.95051/bb:cc:dd:ee:ff:gg/power-level

Event data might or might not be in text:: deciphered by app protocol handlers.

Subscribe

When an app wants to subscribe to events, it can choose to

Receive all events from a particular Class + Optional Filters:
Receive all events from a particular class + app_type
Receive all events from a particular class + app_type + region
Receive specific events from a particular class + app-type
Receive specific events from a particular class + app-type + region

■ Flash Time

Consider a MAC-ID Layer 2 network with all ports and clients (intra-node and external) assigned a UUID.
Example: For Meshdynamics nodes that may be **00:12:CE:00:00:00** through **00:12:CE:99:99:FF** where

00:12:CE the Vendor ID (fixed for that vendor)
: **00:00** to :**99:99** range of MAC-IDs operating within one logical L2 network = 10^4 or 10,000.
: 00 to : FF reserved for the node resident ports (e.g. radios, VLAN, Apps)

Exemplary further segmentation of node resident ports and application hierarchy

Physical Ports are mapped to	: 00 to : 0F	(16)
VLAN Ports	: 10 to : 1F	(16)
Tier 0 Application Framework and Management Apps/Ports	: 20 to :3F	(32)
Tier 1 e.g. Real time sensor streams	: 40 to :7F	(64)
Tier 2 e.g. Aggregation Streams contingent on Tier 1	: 80 to :9F	(32)

Tier Priority: 0,1,2 and map to "flight plans" specific to regional and global streams.
Flight plans and schedules computed to match effective pipe thickness (route capacity).

■ Initialization When an APP is rebooted/reset it will sent

APPID =	*20-2F or 30-3F.*
Event-type	*00 –reset, factory default*
	01 – registration request →
	02 – registration completed ←
	03 – request strategy (only root node knows full tree and congestion maps)
	04 - receive strategy (Hub, or broker)
	05 – request list subscriber
	06 – get list from root.
	07 – send out unicast stream to specific MAC-ID (from list sent 06)
/Event-data	Specific to App and documented in its registration profile.

■ Run Time Strategies (Hierarchy based List or IFFTT rules)

Resident in strategies library

Hub-Spoke	– one to many subscriber list
Broker	- Send to app-root/app-hub node. Let it manage subscriber delivery.
Collaborative	- Distributed, accesses routing table and app needs specs, generates collaborative "flight plans".