ALLSEEN ALLIANCE

Technical Steering Meeting

July 08, 2014

Antitrust Compliance Notice

- AllSeen Alliance meetings involve participation by industry competitors, and it is the intention of AllSeen Alliance to conduct all of its activities in accordance with applicable antitrust and competition laws. It is therefore extremely important that attendees adhere to meeting agendas, and be aware of and not participate in any activities that are prohibited under applicable US state, federal or foreign antitrust and competition laws.

- Examples of types of actions that are prohibited at AllSeen Alliance meetings and in connection with AllSeen Alliance activities are described in the AllSeen Alliance Antitrust Policy. If you have questions about these matters, please contact your company counsel, or if you are a member of AllSeen Alliance, feel free to contact Lee Gesmer or Andrew Updegrove, of the firm of Gesmer Updegrove LLP, which provides legal counsel to AllSeen Alliance.

Reminder:

This call is being recorded

Agenda

- Approve minute from last meeting
- Vote On Smart Home Service Framework Working Group Proposal
- Vote on AllJoyn Audio Service Project Proposal
- AllJoyn.js Technical Review
- Security Status
- 14.06 System Test Overview

Smart Home Service Framework Working Group Proposal

Jun Zhang

SHSF Objectives

- It is proposed to design and develop smart home service framework based on AllJoyn. The interface between AllJoyn smart home server and AllJoyn smart home client needs to be designed and developed.
  - Appliances centralized management: the AllJoyn smart home server connects home appliances and provides the capability to manage home appliances in a centralized management manner.
  - Centralized security: to share appliances services to be accessed by other users under the protection of authority control through the smart home server.
  - Group control: the smart home server provides the capability to control bunch of home appliances in a group manner.
  - Data collection & logging: aggregate available appliances data and log information on the smart home server.

High Level Plan

- The project is dependent on the 14.06 release of AllJoyn Core and Base Service. The first release is planned for 2014/08. The project will provide the first release as slipstream release and then align with the official release cadence.
- Initial contribution will be a general description of AllJoyn Smart Home Service Framework.

Initial Contributors

- Haier
  - Committers
    - Guodong Xue, Director
    - Milton Wang, Vice Director
    - Jun Zhang, Standard Operation Manager
  - Contributors
    - Zhao Ru, Standard Development Manager
    - Qingsong Bai, Standard Development Manager
- BUPT
  - Yonghua Li, Vice Professor – committer
  - Contributors
    - Linghan Li, Engineer
    - Kun Zheng, Engineer
    - Lei Qi, Engineer

Audio Service Project Proposal vote to approve

Scope of Audio Service Project

- Basic service to allow AllJoyn devices to discover and stream audio from a source device to an audio sink device, e.g., from a wireless doorbell to a speaker
  - Support for discovery of available audio content files
  - Registering a listener for sink information
  - Basic controls for pause, stop, play, volume up/down and mute.
  - Support for retrieving audio metadata
- This proposal is solely for the purpose of moving it into the Base Services Working Group
  - Code already in Allseen Git (multimedia/audit.git).

Project Logistics

- Project will be under the base services working group
- Committers
  - Gerry Rovnick (QCE)
  - Josh Hershberg (QCE)
- Git repo
  - allseen/multimedia/audio.git

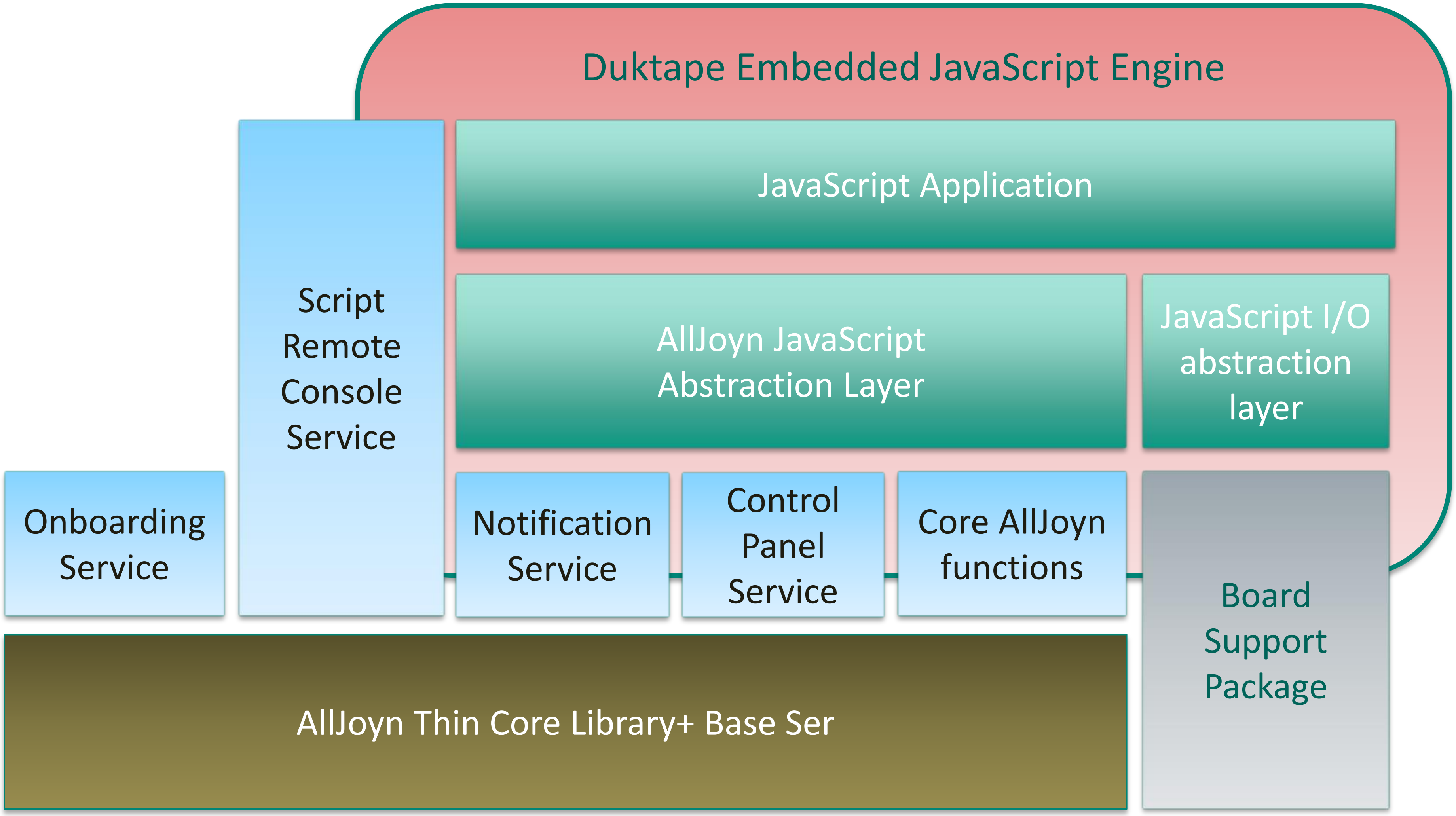
AllJoyn.js Technical Review

Greg Burns
VP, Qualcomm
July 08, 2014

Project Description

- AllJoyn.js combines the AllJoyn thin core library (AJTCL) and base services with a small-footprint ECMAScript 5.0 compliant runtime engine.
- A set of JavaScript APIs provide an easy to use abstraction layer over the AllJoyn core, base services, and the device IO peripherals.
- Combined implementation targeted at microcontrollers having a minimum of128K RAM and 500K flash
  - Will also run on Linux and Windows.

Alljoyn.js Architecture


Duktape Embedded JavaScript Engine
JavaScript Application
Script Remote Console Service
AllJoyn JavaScript Abstraction Layer
JavaScript I/O abstraction layer
Onboarding Service
Notification Service
Control Panel Service
Core AllJoyn functions
Board Support Package
AllJoyn Thin Core Library+ Base Ser

Console Service

- Provides remote access to running JavaScript application
  - OTA flashing of new JavaScript application
  - Execute JavaScript code in real-time
  - Logging of output from JavaScript print and alert functions
  - Displays notifications from JavaScript program
- Plan is to evolve into a full debug interface
  - Breakpoints
  - Single step execution
  - Stack trace
  - Exception logging

LED blinker sample code

```
var cp = AJ.controlPanel();
var c1 = cp.containerWidget(cp.VERTICAL, cp.HORIZONTAL);
var rate = c1.propertyWidget(cp.SLIDER, 500, "Flash rate:");

rate.range = { min:20, max:1000, increment:50, units:"milliseconds" };

var led = IO.digitalOut(IO.pin1);
var blinky = setInterval(function(){led.toggle();}, rate.value);

rate.onValueChanged = function(val) { resetInterval(blinky, val); }

AJ.onAttach = function() { cp.load(); }
```

Doorbell – button push side

```
AJ.interfaceDefinition['org.allseen.DoorBell'] = {
    ding_dong:{ type:AJ.SIGNAL }
};

AJ.objectDefinition['/pushbutton'] = {interfaces:['org.allseen.DoorBell']};

var dingdong = AJ.signal('/pushbutton', 'org.allseen.DoorBell', 'ding_dong');

var pb=IO.digitalIn(IO.pin9, IO.pullDown);

AJ.onAttach = function()
{
    pb.setTrigger(IO.fallingEdge, function(){dingdong.emit(); });
}

AJ.onDetach = function()
{
    pb.setTrigger(IO.disable);
}
```

Doorbell – ding-dong side

```
AJ.interfaceDefinition['org.allseen.DoorBell'] = {
   ding_dong:{ type:AJ.SIGNAL }
};

AJ.onAttach = function(msg)
{
    AJ.addMatch('org.allseen.DoorBell', 'ding_dong');
}

AJ.onSignal = function(msg)
{
    if (msg.member == 'ding_dong') {
        IO.system('aplay DoorBell.wav');
    }
}
```

Notifications – 2 different ones

```
var pbA = IO.digitalIn(IO.pin9, IO.pullDown);
var pbB = IO.digitalIn(IO.pin10, IO.pullDown);

AJ.onAttach = function()
{
    pbA.setTrigger(IO.fallingEdge, function(){
        AJ.notification(1, "Button A pressed").emit(200);
    })
    pbB.setTrigger(IO.fallingEdge, function(){
        AJ.notification(2, "Button B pressed").emit(200);
    })
}

AJ.onDetach = function()
{
    pbA.setTrigger(IO.disable);
    pbB.setTrigger(IO.disable);
}
```

AllJoyn™ Security 2 Architecture

Phil Nguyen
Qualcomm Connected Experiences, Inc.
July 7, 2014

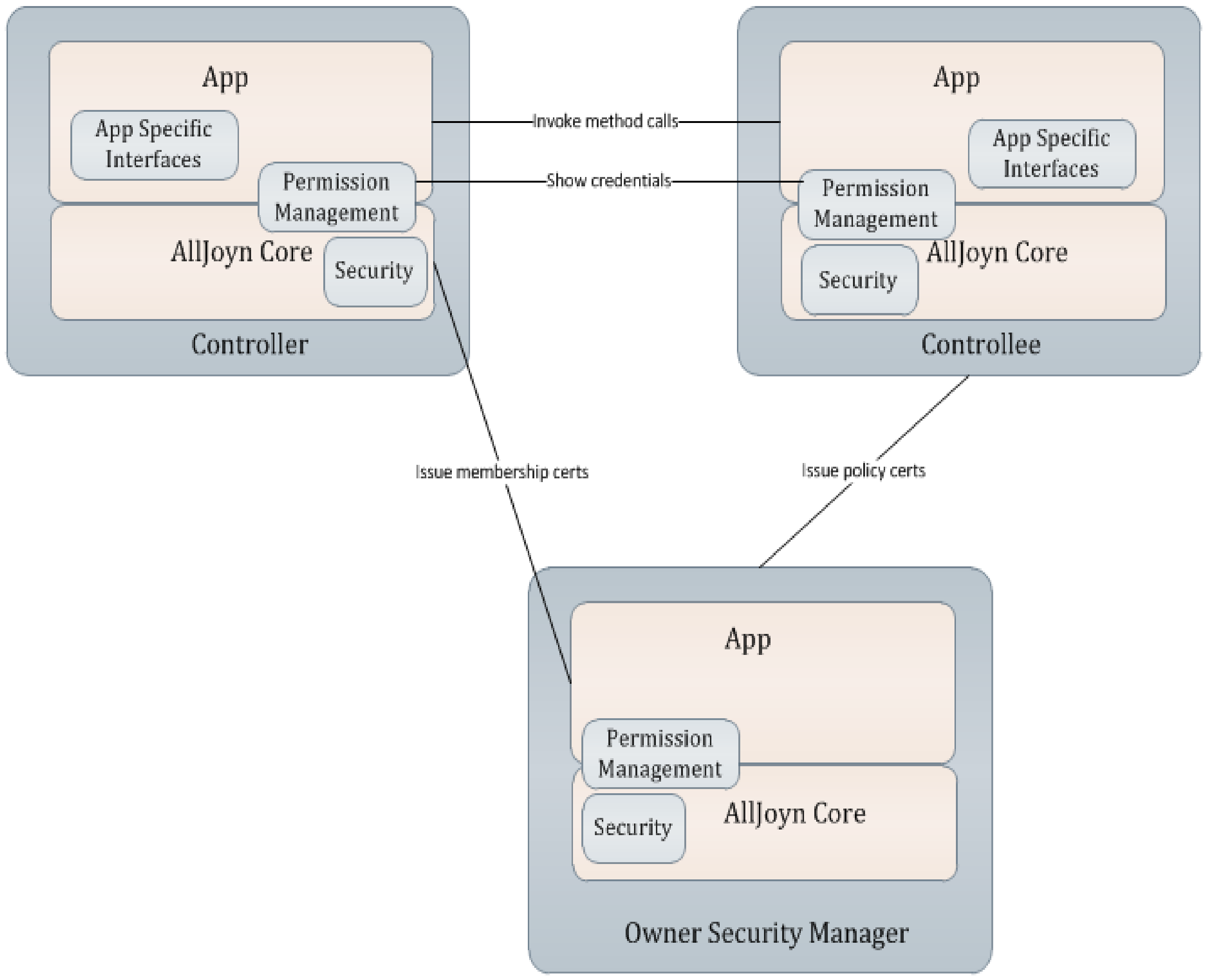
Basic Goals

- Allow end-user to define the access policies to control the access of secure interfaces and secure objects
- Access control is administered and enforced outside of application

Major Components

- Core security
  - Key exchange
  - Encryption
  - Key material storage
- Permission Management Library
  - Manage the permission database
  - Enforce the permission rules
- Permission Manager Utility
  - Utility and interfaces to support certificate management, delivery and installation.

Architecture Update



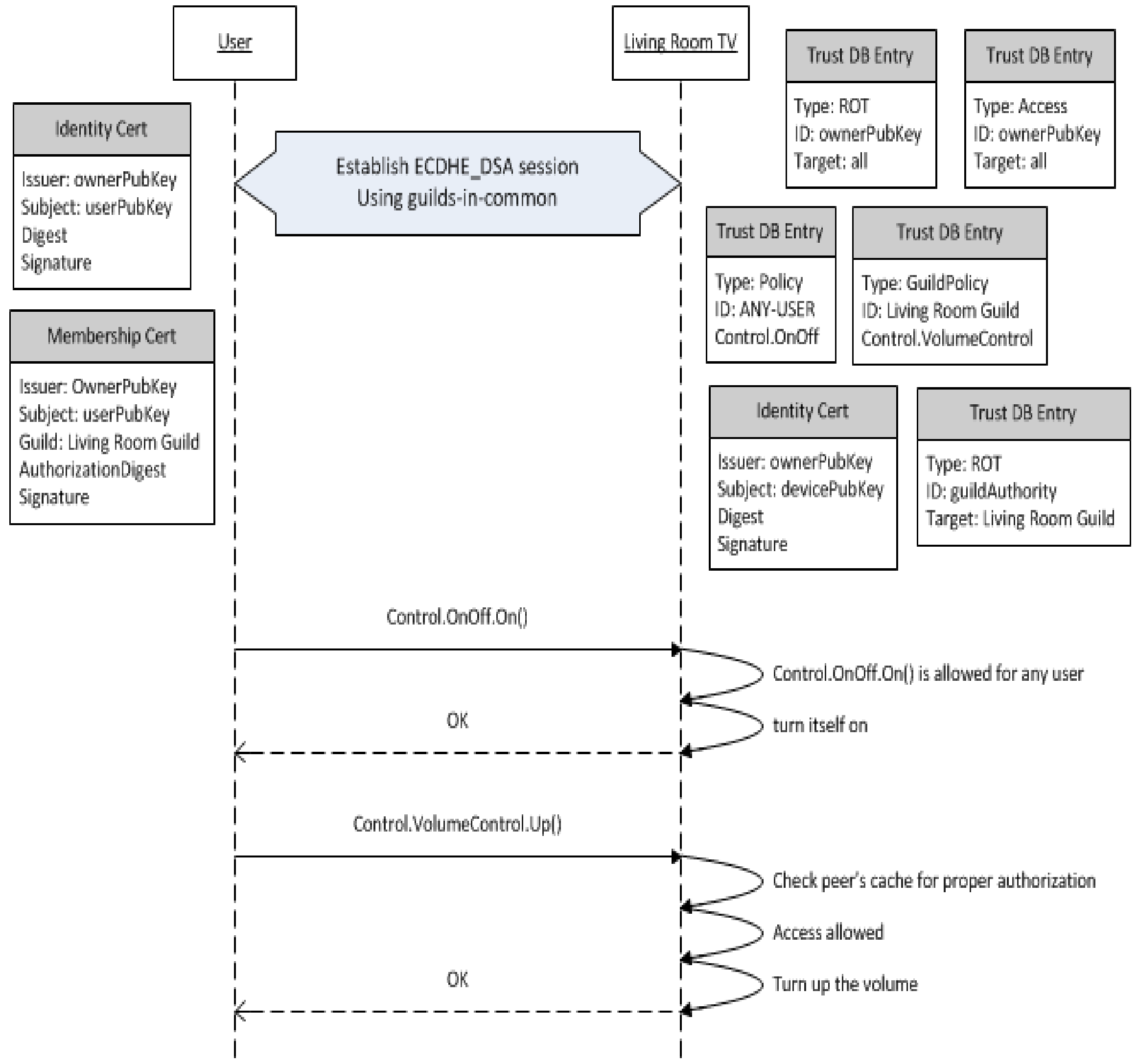
Building Blocks

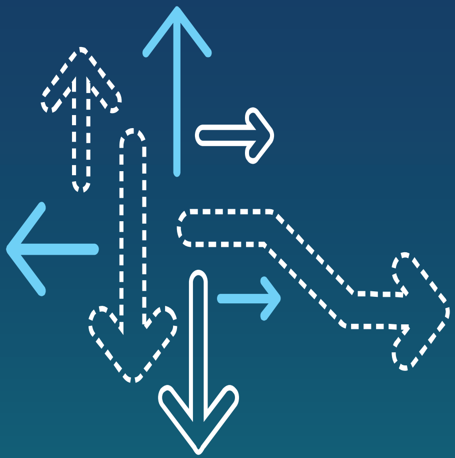
- Encrypted channel with share secret generated using ECDHE key exchanges (already available in AllJoyn 14.06 release)
  - ECDHE_NULL (when claiming a new device)
  - ECDHE_PSK
  - ECDHE_ECDSA
- Guild is a collection of peers that interact with each other in a controlled manner
- Controllee's behavior toward a controller is based on
  - Installed access policy signed by the controllee's root of trust
  - Guild Membership certificates assigned to the controller
- A peer in ECDHE_ECDSA encrypted channel is identified by the public key associated with the encrypted channel

Permission Enforcement Axiom

- The message recipient enforces the permission rules.
  - This means the method call recipient or a signal recipient will enforce the rules.
  - Whenever an encrypted message is received, a callback to the permission library to check whether the given message is authorized for the sender

Typical Validation Flow





14.06
System Test

Objectives

- To characterize the system from user's perspective in a test environment with multiple devices
- To characterize the end-to-end system latency performance for Notification Service under various load conditions (low, med, high) in real environment
- To characterize and benchmark the system stability under various Notifications and dismiss load conditions based on the Basic and Load defined topologies/configurations.
- To validate the testing tools and TestBed setups
- **Automation:**
- Provisioning tools, Test apps for Notification rates, Logging analyzers, Traffic generators.

Test approach and methodology

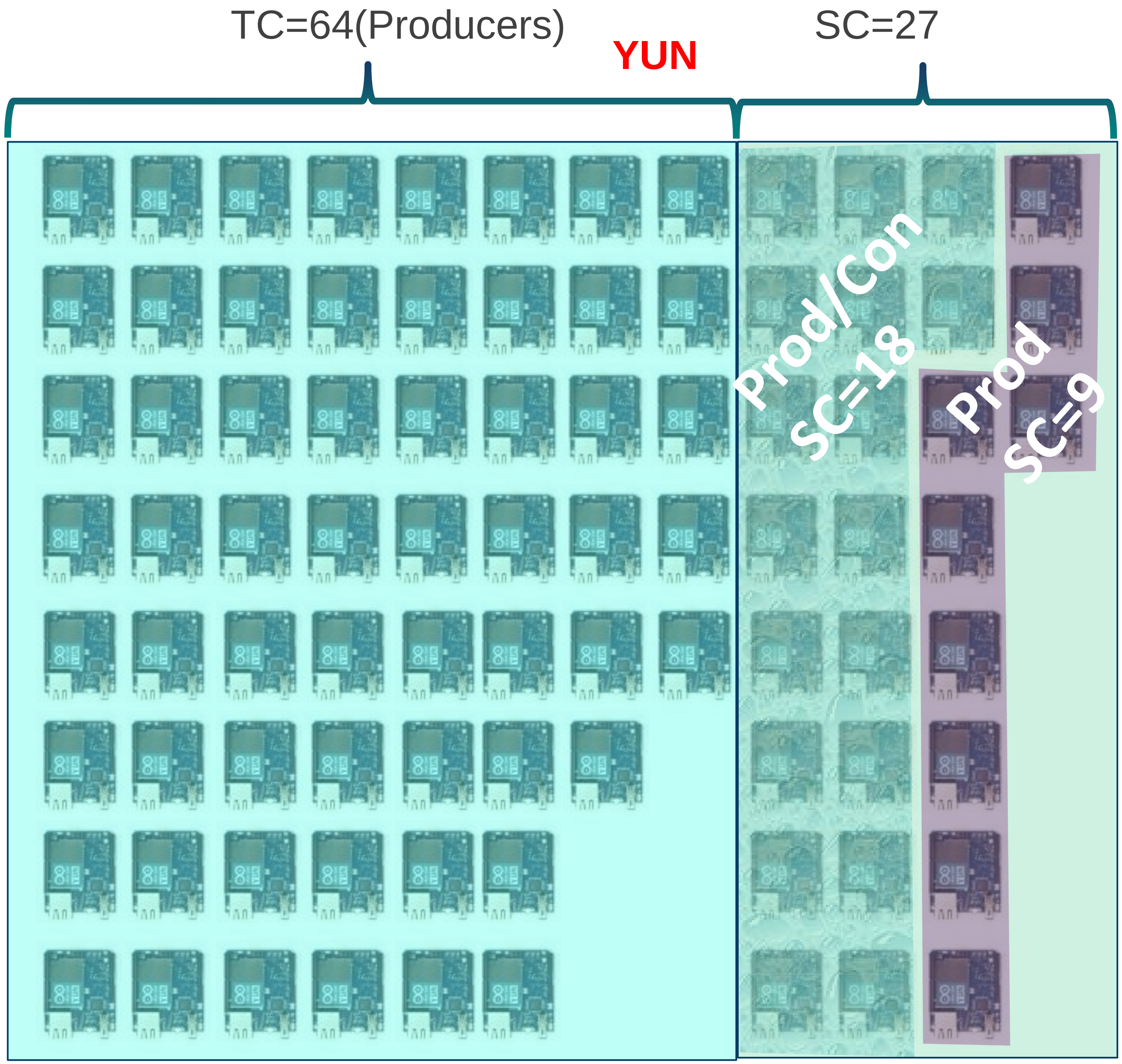
- Performance Testing: Verifies that the system meets:
- The end-to-end Notification and Dismiss performance requirements with specified system topology/configuration by successfully delivering a specified
  - notification message rate from the producers to consumers.
  - dismiss message rate from consumers to producers
- Different variations of load rate will be used

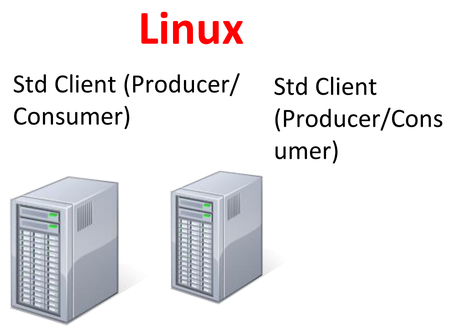
- Stability/Availability Testing: Verifies and characterizes the system stability over an extended period under nominal and load conditions. This test is intended to help identify problems such as memory leaks and flow control processing, etc.

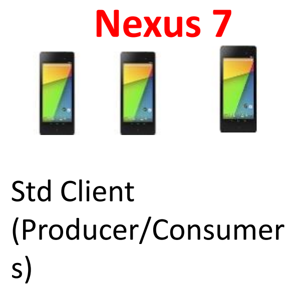
System Test Exit Criteria

- ***Stability and Performance:*** The Notifications service and dismiss feature are characterized. The collected data will be used to obtain metrics that characterize the Latency and the succeed/failed rate of the Notification and dismiss messages. This will be performed in the system test lab environment.
- ***System Test Report***: System test team specified in the test report, detailing the performance/stability of the Notification Service test results.
- ***Test Case:*** Execution Completion of Priority 1 test cases
- ***Defects:*** No P1 and P2 open Bugs. P3 bugs (to be discussed on a case by case basis)

NS-BASIC-01 Virtual Alljoyn Devices (VAD)




Linux
Std Client (Producer/Consumer)
Std Client (Producer/Consumer)



14.06 Status

14.06 Status

- Current status
  - Since 14.06 Beta release
    - 7 issues identified and assigned to v14.06
      - 6 are P2 priority
      - 5 issues closed
    - All issues can be viewed at https://jira.allseenalliance.org
- Tests and characterizations
  - NGNS Cache Refresh
    - Scheduled to be completed by COB 7/8
  - System stability
  - NGNS Cache Refresh
  - NGNS stress and performance
  - AllJoyn ON
  - Services testing on QSDK targets
- AllJoyn v14.06 final targeted for early August
  - Team is currently on track

Technical Steering Meeting

Thank You.