

TECHNICAL AND OPERATIONAL VIABILITY OF COAM DEVICES

AS THE PRIMARY VIDEO SCREEN IN THE HOME

A Technical Paper prepared for SCTE/ISBE by

Matt Kalman

Principal Consultant

IBB Consulting Group LLC

1628 JFK Blvd., Suite 1701, Philadelphia, PA 19103

P: +1 201 658 0623

mattk@ibbconsulting.com

Navinan Rajeswaran, IBB Consulting Group LLC

Geoff Pangman, IBB Consulting Group LLC

Ben Eby, IBB Consulting Group LLC

Max Bamberger, IBB Consulting Group LLC

Table of Contents

Title	Page Number
Introduction	4
Content	4
1. Introduction	4
1.1. Assumptions and scenarios considered	4
1.2. Legacy STB environment as baseline	6
2. COAM apps developed by MSO's for the primary screen	7
2.1. Develop	9
2.1.1. Ecosystems considered	9
2.1.2. Potential for modularity and functional re-use across COAM ecosystems	10
2.1.3. API's / services implications for COAM	14
2.2. Deploy	18
2.2.1. End to end testing	19
2.2.2. App certification and publishing	19
2.2.3. Firmware / OS release schedule	20
2.3. Operate	20
2.3.1. A paradigm shift towards self-care	20
2.3.2. Using app-level and log data	20
2.3.3. In-app Wi-Fi diagnostics	22
2.3.4. Provisioning and authentication considerations	23
2.3.5. Higher stakes for theft of service	23
3. 3rd party COAM apps (per FCC mandate)	24
3.1. Develop	24
3.2. Deploy	25
3.3. Operate	27
Conclusion	28
Abbreviations	29
Bibliography & References	29
Appendix	29

List of Figures

Title	Page Number
Figure 1 - Three scenarios considered	5
Figure 2 - Assumed architecture for analysis	6
Figure 3 - Lifecycle	6
Figure 4 - Total app developers	8
Figure 5 - Most popular streaming devices	9
Figure 6 - 2014-15 streaming market share	9
Figure 7 - Player abstraction	11
Figure 8 - HTML5 abstraction layer	12
Figure 9 - Android OS fragmentation	13

Figure 10 - Blend of native vs. web (circa 2014)	14
Figure 11 - MVPD IP Video Back End	15
Figure 12 - Back end abstraction layer	15
Figure 13 - Cloud enablers for COAM	16
Figure 14 - Netflix dynamic scripting approach	17
Figure 15 - In-app Diagnostics	21
Figure 16 - COAM Population Health Monitoring	22
Figure 17 - 3rd Party COAM Apps	24
Figure 18 - Player Abstraction For Third Party Apps	25
Figure 19 - Summary of Deployment Challenges with FCC Mandate	27
Figure 20 - Summary of Operational Challenges with FCC Mandate	27
Figure 21 - Traditional SOA vs. Micro-services	30

List of Tables

Title	Page Number
Table 1 - Deployment Control Comparison	18
Table 2 - Deployment Model per FCC Mandate	26
Table 3 - Summary Comparison of MVPD v. 3rd Party Apps	28

Introduction

The traditional Multichannel Video Provider (MVPD) video business has seen a world of change. Programming costs are higher than ever, customers are supplementing subscriptions with OTT services and migration to IP efforts are being evaluated to target better efficiencies as new video formats introduce heavy bandwidth constraints alongside improved picture quality.

Despite these changing market dynamics, the cost of QAM-based video customer premise equipment has not declined significantly. Further, the way subscribers consume this content – via leased set top boxes – persists. Potential regulation and a shift to IP-based devices may finally change this. The question for MVPDs is whether a Customer Owned and Managed (COAM) device future is actually possible.

It should be noted that MVPDs started the journey towards COAM several years ago. In 2011, Cablevision's Optimum app let subscribers access content via iOS devices. Similarly, Time Warner Cable launched its TWC TV on the Roku platform nearly four years ago and most recently doubled down on this approach by making a push toward a full linear lineup with Title VI compliance (e.g. with emergency alerts and closed captioning) in its New York trial.

However, MVPD's may no longer have the luxury of moving at its own pace. The recent FCC "Unlock the Box" ruling raises the stakes and operational risk profile for COAM by mandating MVPD support for third-party developed applications that would put a new demarcation between a subscriber's video service and the app they use to access it. It remains to be seen how the "Unlock The Box" initiative will unfold. In the meantime, operators are beginning to explore the feasibility of an IP-based COAM devices in lieu of the traditional leased STB model.

Content

1. Introduction

1.1. Assumptions and scenarios considered

A few key level-setting points around the scope of this paper

1. We are assuming a Title VI compliant video service at parity with a cable STB in the home
 - Full linear channel lineup inclusive of all local and PEG channels
 - Emergency alerts (EAS/EAN messaging)
 - Closed captioning
 - Local ad insertion / no change to media sales model
2. The key actors in this paper are:
 - The subscriber
 - The MVPD – the traditional Title VI video service provider, who maintains the business relationship with the subscriber
 - The COAM device manufacturer
 - The COAM app ecosystem owner
 - i. NOTE: these are often the same as the device manufacturer, but not always (e.g. Smart TVs use an Android application ecosystem)

- The app owner
 - i. NOTE: this is the MVPD with the exception of the FCC's unlock the box use case, where it is a 3rd party (e.g. COAM device manufacturer, software developer, etc.)

Given those assumptions, this paper will consider three scenarios:

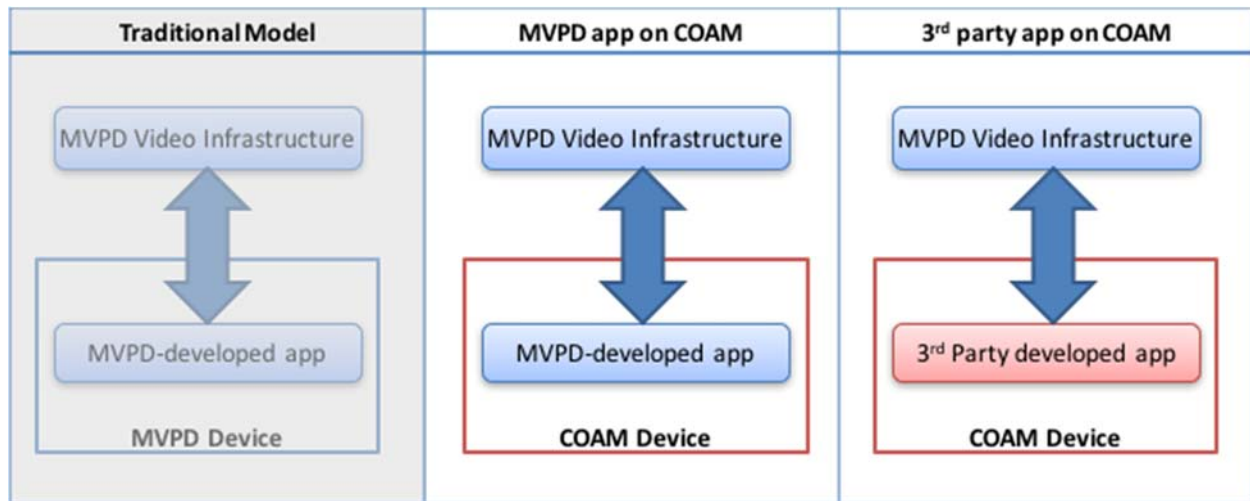


Figure 1 - Three scenarios considered

- **Traditional model:** MVPD STB via MVPD managed network over QAM, IP or hybrid for video delivery. This is considered in the paper to frame the discussion for COAM and does not dwell on legacy COAM models using CableCard technology such as TiVO.
- **MVPD COAM model:** MVPD develops apps for portfolio of COAM devices. NOTE: the paper will focus its attention primarily on this scenario.
- **FCC Mandate:** 3rd party developed app on COAM devices for consuming both MVPD content

NOTE: although highly relevant, consumption of third party content in either the MVPD COAM model or FCC mandated approach is out of scope. This analysis focuses on consumption models for MVPD content.

While many possible architectures may be considered, this white paper assumes the following:

1. The end-to-end video path is "unmanaged" IP video or OTT and delivered from the cloud.
2. Recordable content is supported, but only in the form of cDVR. Recordable output requirements are not addressed (e.g. CVP2).
3. The gateway is HSD only – there are no video adaptation functions, nor is multicast to unicast translation assumed. Content is presented as an OTT unicast stream to the consuming COAM device. There is no consideration of DLNA or Vidipath distribution in this paper.
4. Home LAN connectivity is Wi-Fi. Distribution of content via MoCA and Ethernet is plausible, but not in scope for this analysis.

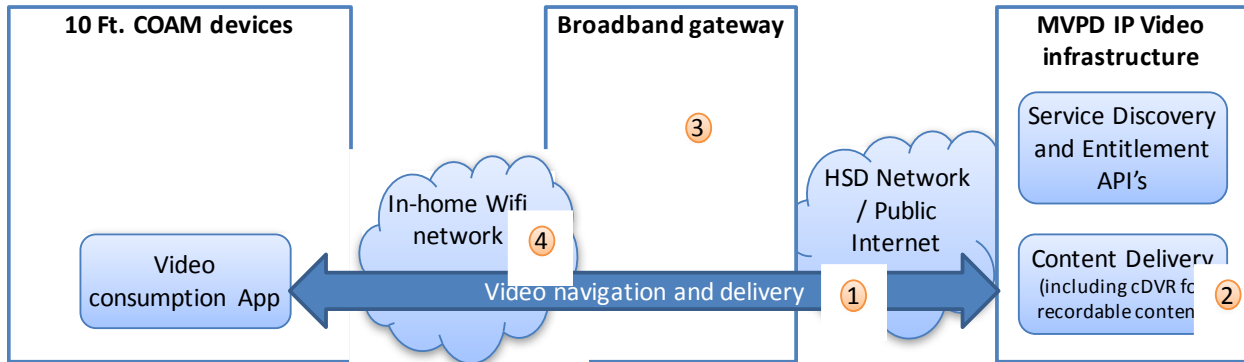


Figure 2 - Assumed architecture for analysis

Finally, this analysis will be based on the lifecycle of COAM applications and discuss implications across it:

- **Develop:** the engineering behind COAM applications
- **Deploy:** the operational process for certifying COAM applications and making them available to customers
- **Operate:** Ensuring that COAM applications in the field can be supported both in terms of single customer troubleshooting and monitoring/resolving global issues

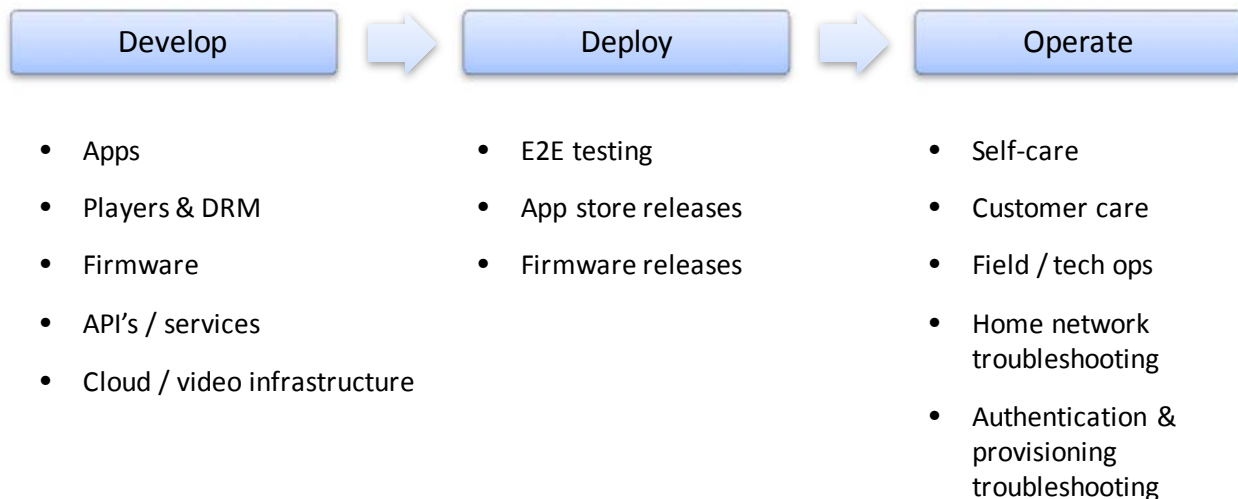


Figure 3 - Lifecycle

1.2. Legacy STB environment as baseline

Historically, when it comes to the subscriber video experience, MVPDs are used to having full control over all aspects of the software, firmware, and infrastructure lifecycles toward delivering video content to subscribers. STB code, whether developed internally or in conjunction with the vendor community, are certified within MVPD labs and deployed according to a schedule and platform that MVPD's have full control over. Additionally, MVPDs have the freedom to further drive the customer experience by owning and maintaining control and visibility of the service delivery path.

The development environment of legacy set-tops is a closed, proprietary ecosystem. Yet it is one known by many in the industry and one that MSO developers have grown comfortable with:

- Front end development consists of authoring navigation apps (namely the guide) against the OCAP or RDK-V specification implemented by the vendor community or working directly with those vendors on implementing software against their proprietary software/firmware stacks
- Back end development depends on which generation of guide
 - o More recent MVPD guides and applications are either cloud-based or at least web-enabled and therefore share the same or similar API's that are used by their second screen applications. These API's are often located in the cloud, national or regional data centers.
 - o Older legacy guides rely strictly on information passed through headend infrastructure
- End to end testing is truly end to end and controlled completely within the lab environment. All elements of a release are tested against one another.
- Pushing code to STB's and back end services in MVPD's strictly defined operational maintenance windows.

With a high-degree of control over the STB software stack, MVPD's dictate the degree of visibility required for remote troubleshooting by customer care or higher tiers of operations. This visibility is typically implemented through an SNMP or TR-069 data model exposing real time diagnostic information – this information can be used to identify if there is a specific issue with customer premise equipment or if there are systemic issues in the network. Increasingly, these same diagnostics are made available to customers through self-care portals. Customer care issues a truck roll if customer troubleshooting is not successful remotely.

Systemic issues are escalated throughout the traditional tiered model:

- **Tier 1:** Customer care agents are made aware for systemic issues by higher tiers of operations (e.g. through NOC notifications). Customer care can also escalate issues that may be systemic to a NOC or using a bridge model.
- **Tier 2:** NOC's are proactively monitoring all aspects of the end to end video solution and can escalate to higher tiers of operations accordingly.
- **Tier 3:** Higher tiers of operations (or DevOps) are using tools for monitoring the end to end solution and proactively actively managing production issues such as capacity limitations.
- **Tier 4+:** Engineering teams are engaged for bugs or defects that are found in production.

Finally, it should be noted that the traditional STB model can solve for IPTV scaling – e.g. if there is a calculated decision to shift all leased CPE to IP based video delivery. It is therefore not within the scope of the COAM analysis to solve for scaling the back end infrastructure or the network (e.g. Content Delivery Network) to accommodate an all IP end game.

2. COAM apps developed by MSO's for the primary screen

This section assesses what it would mean to have a COAM device as a true STB replacement. It should be acknowledged that many of these opportunities and challenges are already being addressed by using COAM as a second screen. However, second screen comes with a different set of expectations – we assume here that customer expectations are similar to the traditional STB environment outlined above and what the MSO can do to address those expectations across the lifecycle.

This section also does not seek to make economic arguments for or against COAM. Therefore, the economic tradeoffs of shedding STB CAPEX by using COAM while losing STB and A/O revenue generated by the MSO's should be considered a wash.

The benefits of using COAM ecosystems for user experience development are apparent. The openness and flexibility of the SDK's allow MVPD's to create rich, immersive experiences on par or better than their OTT competitors. Finding the talent and workforce to create these experiences is far easier than legacy:

Total Number of Developers by App Store

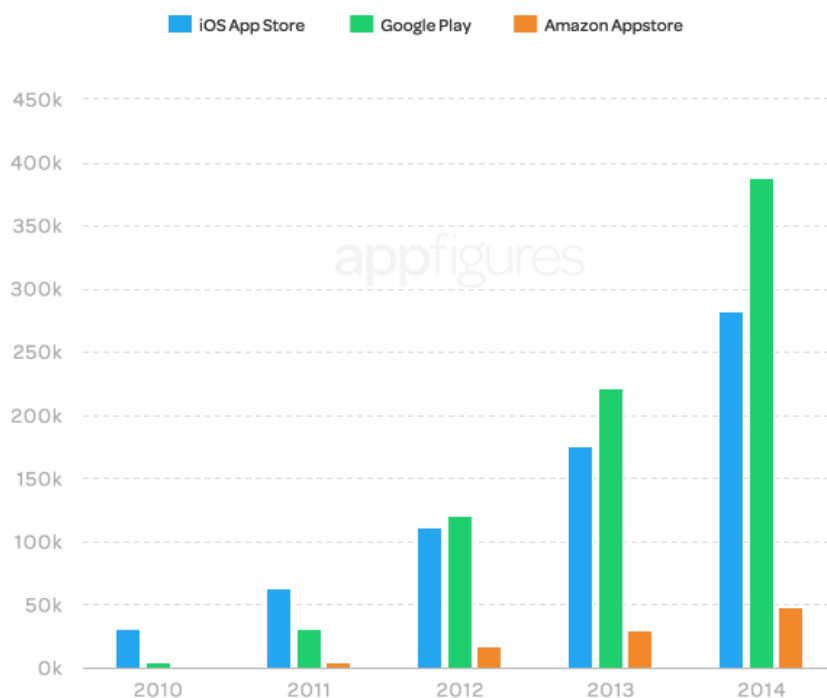


Figure 4 - Total app developers

<http://blog.appfigures.com/app-stores-growth-accelerates-in-2014/>

Furthermore, the Bureau of Labor Statistics estimates the number of web developers in the US alone at close to 150k (<http://www.bls.gov/ooh/computer-and-information-technology/web-developers.htm>). These figures are impressive as of year-end 2014 and we can may be able to assume they have grown significantly since.

2.1. Develop

2.1.1. Ecosystems considered

For the purposes of simplicity, we are using recent research from Parks Associates to segment COAM devices in this analysis:

2015 U.S. Streaming Media Player Sales
 U.S. Broadband Households that Purchased a Streaming Media Player

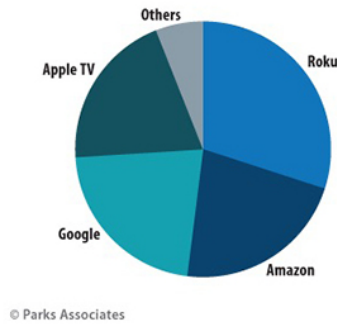


Figure 5 - Most popular streaming devices

<http://www.parksassociates.com/blog/article/pr-05172016>

Evidence is corroborated by research from strategy analytics for the past two years:

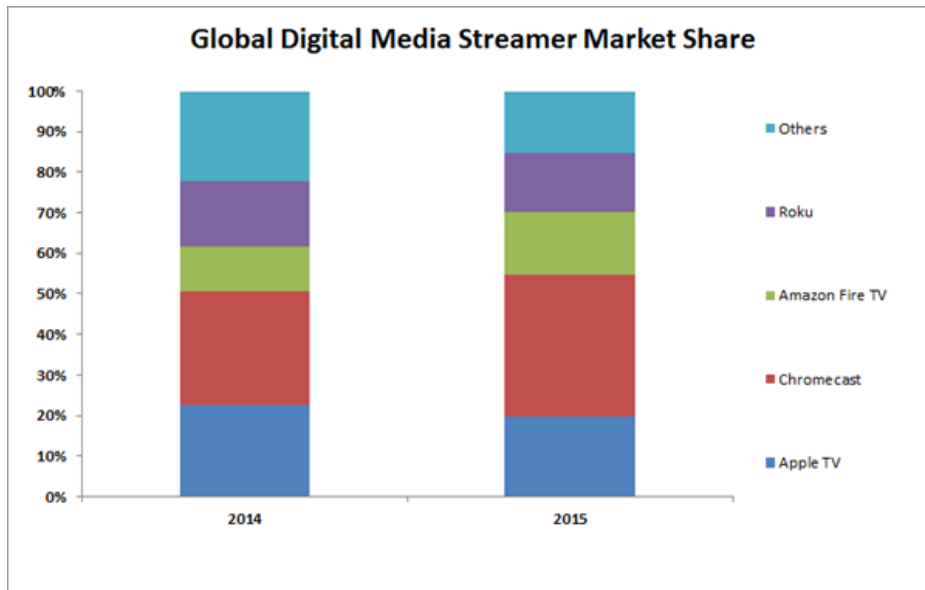


Figure 6 - 2014-15 streaming market share

<https://www.strategyanalytics.com/strategy-analytics/news/strategy-analytics-press-releases/strategy-analytics-press-release/2016/03/08/chromecast-takes-35-of-the-42-million-unit-global-digital-media-streamer-market-in-2015-says-strategy-analytics#.V4FV4VcbwU1>

Therefore, this analysis makes the following assumptions / confluations of device ecosystems in the scope of this paper

- Roku
- Android and Android TV
- Amazon Fire
- iOS / AppleTV
- HTML5 / Other

NOTE: We are considering Chromecast an extension of iOS and Android as those are the dominant mobile devices on which the MVPD app would reside.

2.1.2. Potential for modularity and functional re-use across COAM ecosystems

One potential challenge of a COAM end game is the notion of ongoing bespoke development by ecosystem at scale. To date, the notion of develop once, run everywhere has been unworkable – MVPD's have needed to author applications for each platform they want to run on.

2.1.2.1. HTML5 approach

For years MVPD's have been looking to HTML5 video in lieu of proprietary and bespoke development as a possibility for re-use of functionality across devices. Early years of HTML5 were challenged by poor performance issues, browser dependency, and low adoption.

However recent trends suggest this approach may be closer to reality. Specifically, non-browser support using a Web Views model within iOS and Android SDK's may be the most telling:

<https://www.infoq.com/news/2014/11/AndroidiOSHTML5>.

- Android's Lollipop release includes Updatable Web Views which brings Web Views to parity with Chrome/Chromium browser capability: <https://infinum.co/the-capsized-eight/articles/the-updateable-webview-on-android-5-lollipop-what-is-it-and-why-should-you-care>
 - Amazon Fire also supports Web Views in conjunction with Android SDK - <https://developer.amazon.com/public/solutions/platforms/android-fireos/docs/building-and-testing-your-hybrid-app>
- Apple significantly upgraded HTML5 Web View performance between its iOS7 and iOS 8 releases, moving from UI Web View to WK Web View. Some performance improvements documented here: <https://www.sencha.com/blog/apple-shows-love-for-html5-with-ios-8/>

If development organizations can re-use web apps (HTML5, CSS and Javascript) across iOS, Android and browser environments, this analysis contends that a high degree of code re-use is feasible given above

statistics and trends (inclusive of iOS and Android apps using Chromecast). Furthermore, recent trends suggest that an HTML5 hybrid web app approach is becoming mainstream: <http://blog.venturepact.com/8-high-performance-apps-you-never-knew-were-hybrid/>. And consortiums such as Apache Cordova are jumping in to standardize how web app code can be re-used across platforms (<http://cordova.apache.org>).

2.1.2.1.1. Player abstraction

A number of MVPD's are also beginning to abstract player development from the overall app and user experience. This has an added benefit for functionality and code re-use and consistent video playback experience across different COAM ecosystems. It can also be linked to the aforementioned advances in HTML5, and more specifically HTML5 video.

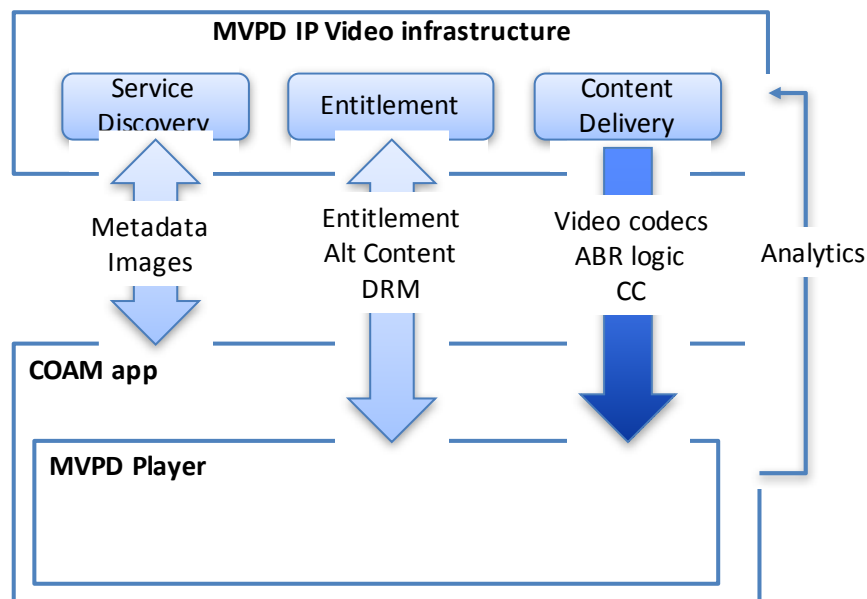


Figure 7 - Player abstraction

Currently, Adobe is still a dominant “player” (pun intended). But with Apple dropping support for Adobe years ago, fragmentation has been inevitable. With HTML5 video support also prevalent, coalescence around a common HTML5 video player is being looked at by the industry as a future standard.

These trends give credibility to the notion of MVPD industry proposals such as “Ditch the box” and DSTAC. The diagram below shows the W3C specifications for HTML5, Encrypted Media Extensions (EME), Media Source Extensions (MSE), and Web Crypto APIs produces a detachment from the underlying hardware, CA/DRM and OS platforms. Using this approach, MVPDs could, in theory, write one HTML5 web app that runs across all platforms that support compliant HTML5, EME, MSE, and Web Crypto.

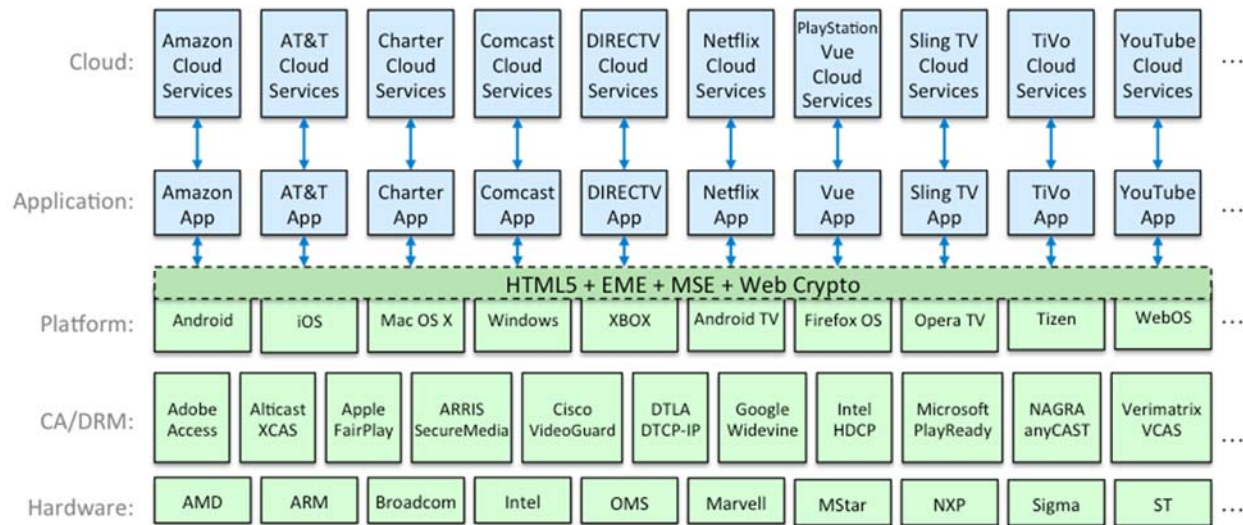


Figure 8 - HTML5 abstraction layer

<http://www.cablelabs.com/downloadable-security-and-the-future-of-cablecards/>

This approach ensures 3rd party devices manufacturers will respect customer privacy, advertising rights, and copyrights. This also protects the technical integrity of the app which will enforce consumer protections, content security, and all of the programming licensing of the MVPD. The HTML5 security proposal, supports multiple DRM systems from Microsoft PlayReady, Adobe Access, Apple FairPlay, as well as others.

2.1.2.1.2. HTML5 – almost, but not quite

The above HTML5 proposal is a noble end game for the industry – MVPD’s and OVD’s alike. However, we believe this transition will take some time, primarily due to hardware and OS fragmentation. For example, as of January this year, Lollipop or later had only been deployed less than a third of Android

devices globally:

Version	Codename	API	Distribution
2.2	Froyo	8	0.2%
2.3.3 - 2.3.7	Gingerbread	10	3.0%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.7%
4.1.x	Jelly Bean	16	9.0%
4.2.x		17	12.2%
4.3		18	3.5%
4.4	KitKat	19	36.1%
5.0	Lollipop	21	16.9%
5.1		22	15.7%
6.0	Marshmallow	23	0.7%

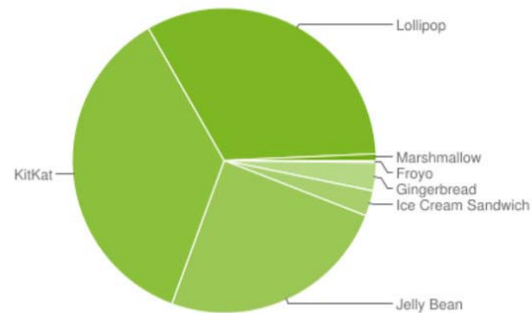


Figure 9 - Android OS fragmentation

<http://www.androidcentral.com/lollipop-now-326-percent-android-devices-marshmallow-07-percent>

Furthermore, there are still performance issues to be ironed out using the Web View approach. After all, a Web View is merely an implementation of the browser within the native app, and performance and reliability issues are still highly relevant. Additionally, there is inevitable customization required for some COAM features that may only be available natively – this can marginalize code re-use potential and might drive an MVPD to still use a purely native implementation rather than hybrid.

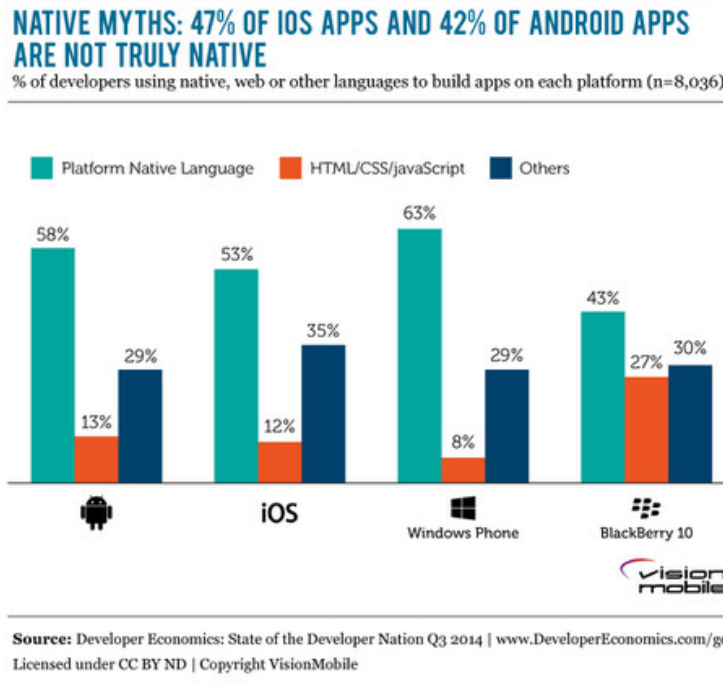


Figure 10 - Blend of native vs. web (circa 2014)

<http://readwrite.com/2014/10/02/html5-apple-ios-8-wkwebview/>

Finally, there are certain ecosystems with significant market scale, such as Roku which offer no support for HTML5 and web applications and have not stated plans openly to do so. The overall point is – common HTML5 code running across the majority of devices will not happen overnight and MVPD’s will need to support bespoke COAM ecosystems for the foreseeable future.

2.1.3. API’s / services implications for COAM

Viewers today want a consistent TV experience across all screens. For MVPDs, this requires the ability to introduce UI changes and new service features across all devices at the same time. In a COAM world with increasing bespoke development this becomes particularly challenging.

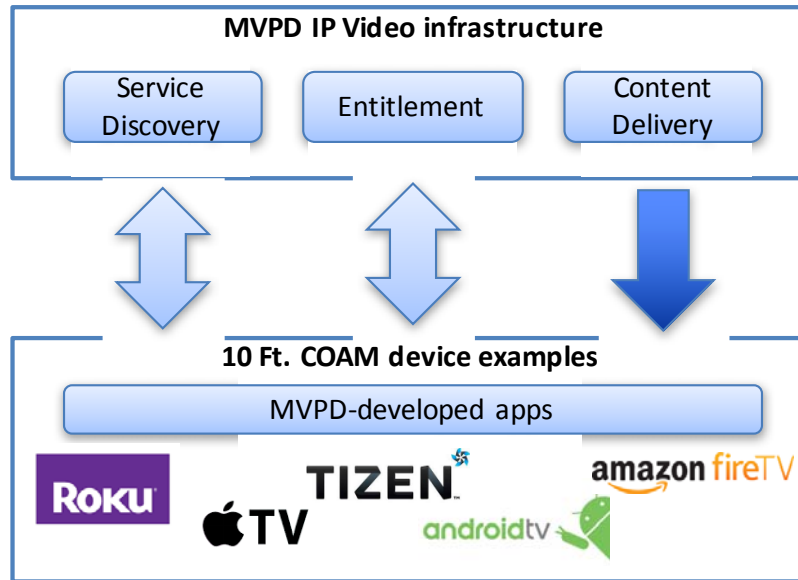


Figure 11 - MVPD IP Video Back End

A number of core services are essential to the MVPD back-end and must be accessible across a large and growing COAM application portfolio. These services can be classified as:

- **Service discovery:** “control channel” data / the channels and programs that the carries in addition to underlying metadata and images for navigation
- **Entitlement:** Whether a consumer has the right to access each of those channels and programs and the usage rights that a consumer has with respect to those channels and programs (e.g. in home vs. out of home). The application of DRM to protect content can also fall into the category.
- **Content Delivery:** The packaging and streaming infrastructure / content delivery network (CDN used to deliver video to the consuming device

Many MVPDs have developed, or are developing, a common back end and using an abstraction layer to simplify integration with legacy STBs and a portfolio of bespoke COAM apps. Through the abstraction layer, changes to the back-end services only need to be developed once to run on all devices.

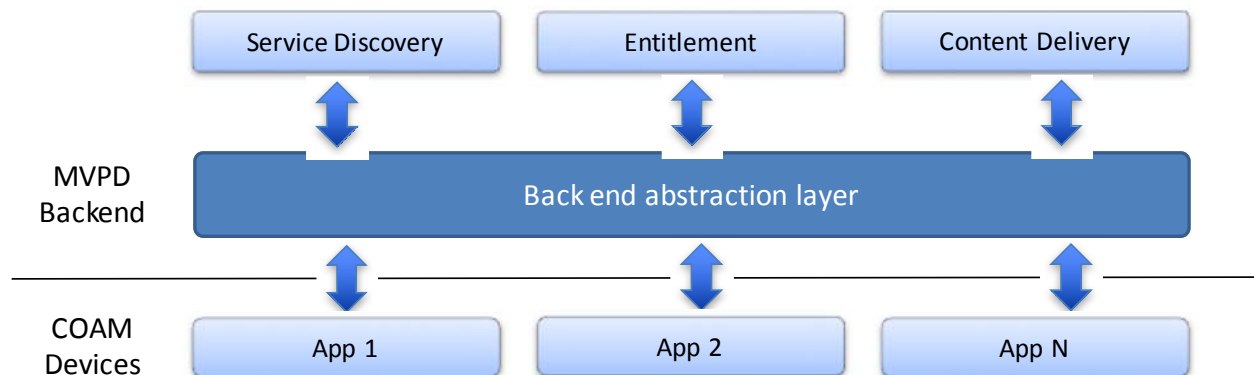


Figure 12 - Back end abstraction layer

However, we see three potential pitfalls with this simplified approach:

1. **Code re-use is not absolute:** As aforementioned, while having a common code base across COAM clients is a worthwhile goal, it is not current reality– each ecosystem will have unique needs and likely a different pace of change.
2. **An abstraction layer can become a bottleneck** – Architecturally, sending all API calls to a single instance can become a single point of failure. API teams can also become a bottleneck form a development perspective.
3. **Monolithic systems on the back end** – Similarly, individual service discover, entitlement and content delivery systems can slow down the pace of innovation and also jeopardize reliability of the ecosystem. For example, if an identity system goes down it can take down the entire API layer and cause an outage for the entirety of the COAM ecosystems.

All of these risks become even more important to address once customers are relying on a wide portfolio of COAM devices as the primary screen for linear viewing.

This analysis suggests that there are two “cloud” enablers for COAM as the primary screen: client-specific adapters and a micro-services API architecture. It should be noted that both are not specific to COAM and that MVPD’s should also consider using these common approaches for next generation or IP based leased devices for further architectural and operational efficiency.

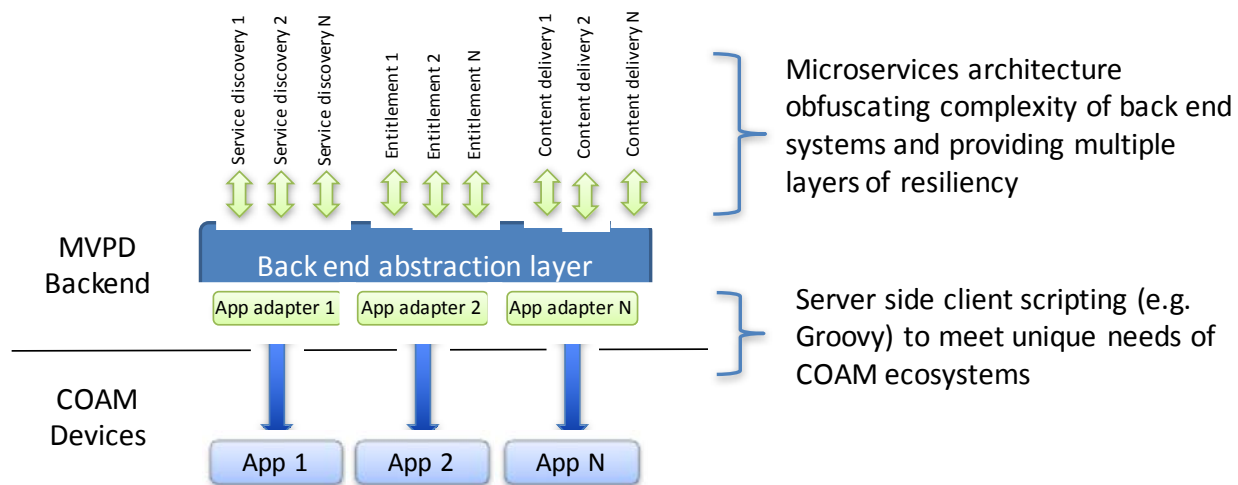


Figure 13 - Cloud enablers for COAM

2.1.3.1.1. Server side scripting

In order to meet the unique demands of individual application ecosystems, MVPD’s may consider implementing individual client adapters as the front door to the back end. In this model, client teams write their own adaption scripts and deployed without tight coupling with the rest of the back end architecture. This approach also distributes risk at the abstraction layer. Netflix has recently championed this approach and their implementation is depicted in the following diagram:

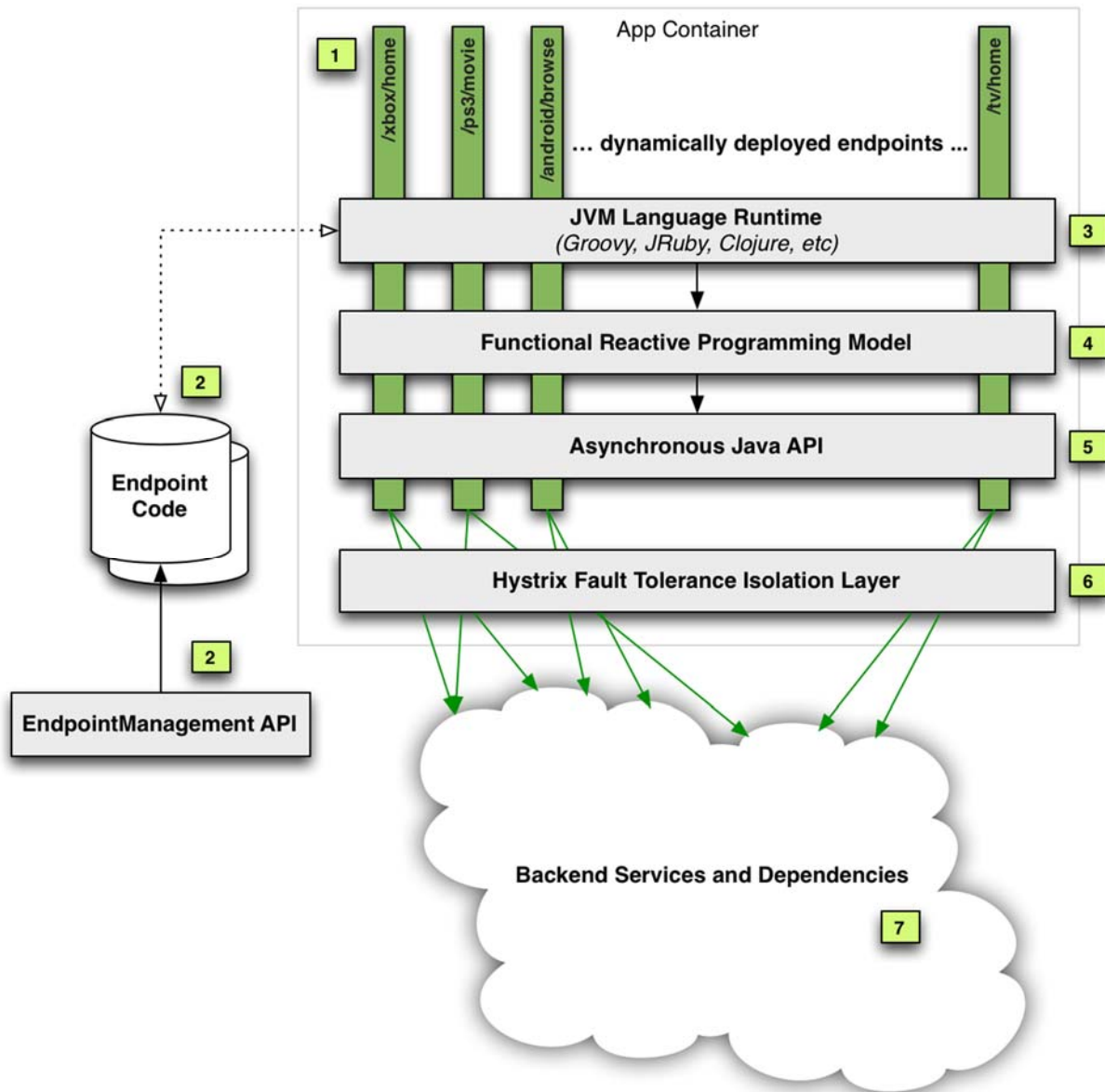


Figure 14 - Netflix dynamic scripting approach

<https://www.infoq.com/news/2013/02/netflix-api-optimization>

2.1.3.1.2. *Micro-services*

Many MVPDs and online video distributors (OVD's) are also choosing to focus on a micro-services architecture composed of small, modular software components to serve a large portfolio of COAM applications. A micro-services architecture allows the service provider to de-couple back-end changes from the device and guide, enabling faster deployments across a portfolio of devices and in turn better control of the user experience. This architectural method is particularly ideal when the service provider

must support a range of platforms and devices, and to deal with the ambiguity of future, yet unknown, COAM devices and apps.

Many of today’s tech giants use micro-services to keep up with rapid changes in the digital world – especially across a myriad of end devices and platforms (think mobile, IOT etc.). Netflix, eBay, Amazon, Twitter, PayPal and many other large-scale websites and applications have all evolved from monolithic closed” applications to a micro-services architecture.

[\(https://smartbear.com/learn/api-design/what-are-microservices/\)](https://smartbear.com/learn/api-design/what-are-microservices/)


Netflix, in particular, has a widespread architecture that leverages micro-services. It receives more than one billion calls every day, from more than 800 different types of devices, to its streaming-video API. Each API call then prompts additional calls to multiple other backend service.

[\(https://smartbear.com/learn/api-design/what-are-microservices/\)](https://smartbear.com/learn/api-design/what-are-microservices/). Here abstraction is particularly important to allow rapid scale and growth at low cost. Additional information on micro-services can be found in the Appendix of this document.

2.2. Deploy

Interestingly, the deployment aspect of COAM applications developed by MSO’s for the primary screen is ultimately not an overly complicated or laborious activity. At its surface, and especially from a typical modern app developer’s point of view, having to contend with deployment constraints of COAM device ecosystems *could* present significant challenges. These challenges can be identified from the simple reason that the MVPD app on a COAM device scenario ostensibly limits the level of visibility & control over a number of areas; i.e. app certification/publishing, app release schedules, COAM firmware releases, and end-to-end testing. The table below represents a surface evaluation of who owns key activities we’ve identified in the app deployment use case:

Table 1 - Deployment Control Comparison

	End-to-end testing	App Certification & Publishing	Firmware / OS Update Schedule
MVPD guide on the MVPD STB	MVPD	MVPD	MVPD
COAM app*	MVPD	3 rd party app ecosystem	COAM device 3 rd party
Impact	Additive 	Different but substitutive	

* developed by MVPD for the COAM primary screen device

With COAM devices deployed as the primary screen (in lieu of the STB) the MVPD’s purview over deployment activities does change significantly.

2.2.1. End to end testing

End to end testing is one aspect of COAM as a primary screen that will most likely add complexity for MVPD's at the front end of the deployment process. As a second screen offering today, there is less of a consumer expectation for feature parity across COAM ecosystems. For example, at some MVPD's local channels are offered on one COAM ecosystem, while they are missing from another. We posit that with COAM as a primary screen this consumer expectation changes considerably. This means that releases must be more carefully coordinated, in turn implying that cross-app testing must be done in parallel.

Further complicating matters, there should be no expectation that leased or MVPD-managed devices go away anytime in the near future. This means that the impact of COAM on testing teams and lab environments is **additive** rather than substitutive. MVPD's will be required to test a large portfolio of COAM applications (and various permutations of firmware and OS builds) in parallel with their own devices.

It is therefore essential to add highly skilled and motivated test resources while in parallel expanding and hardening test environments in preparation for a future of COAM as a primary screen. Many MVPD's are already testing second screen COAM apps in parallel to their leased applications and devices, but as a primary screen, the stakes are higher and the right investments should be made accordingly.

2.2.2. App certification and publishing

Historically, MVPDs have had to coordinate activities in order to manage controlled releases to subscribers. While this improves for MVPD's with leased IP devices, app store certification & publishing will require different expectation of design and performance for approval. This is already being managed by MVPD's who offer second screen video applications; the stakes are simply higher with the primary screen.

The obvious disadvantage in the COAM model (vs. traditional) is a loss of control over when updates are received by large portions of the video subscriber base. However, many of the aforementioned development steps, this risk can be mitigated:

- Using client adaption layers can allow for configuration or feature flipping from the back end infrastructure rather than the app itself
- Use of HTML5, CSS and Java script through Web Views raises the possibility of remote app configuration and multivariate testing without having to go through formal app store approval for all changes

Even without both of the above, there are tools – both purchasable and open sourced – which can be used for remote configuration and multivariate testing within native COAM applications. For example, Clutch.io is an example of a tool that Twitter has open-sourced for use in iOS and Android environments: <http://venturebeat.com/2012/10/11/twitter-open-sources-clutch-io-so-developers-can-easily-add-ab-testing-to-ios-and-android-apps/>

2.2.3. Firmware / OS release schedule

On one hand, MVPD's may view abdication of firmware and OS management as a blessing. For years, MVPD's have been required to tightly manage the coupling of vendor STB stack upgrades alongside guide deployments; many would argue that this process has never been easy or seamless.

There is one potential hiccup is: the possibility of a deployed app becoming error prone due to a firmware or OS update to the COAM device. However, for major app stores such as iOS and Android, backward compatibility between device firmware / OS and deployed applications is a common expectation. With these expectations in place, we believe this risk is somewhat mitigated.

That said, major OS updates should be viewed as additional COAM app permutations. And with the expectation of a primary screen quality viewing experience, it is essential that all new OS and firmware deployments be tested for reverse compatibility. This re-emphasizes the importance of test resources when considering a COAM future, but it also highlights the importance with which MVPD's should prepare for major OS and firmware releases.

2.3. Operate

A move towards COAM as the primary screen will move service providers out of their comfort zone of leased devices which currently use standards such as TR-069 and SNMP for remote troubleshooting. Most COAM devices actually do not allow the MVPD access to any device-level diagnostics and management unless the device is in developer mode. Since this is infeasible, an MVPD may need to rely on application level data for traditional care, or perhaps rely more heavily on self-care. MVPD's should work together and in conjunction with the retail community to agree on a standard that can ensure customer support will remain seamless when a customer opts for a COAM device.

2.3.1. A paradigm shift towards self-care

In general, MVPD's have a great opportunity to lighten customer call and truck roll volumes by moving towards a COAM model. By definition, COAM devices are designed to be self-install and often are much more intuitive than the installation and provisioning process associated with a STB – which often requires the assistance of a field technician. This entire process is outside the purview of the MVPD, who is no longer operationally accountable for it.

However, the MVPD is unlikely to abdicate responsibility for customer troubleshooting. It can, however, use rich application data available from COAM ecosystems to further promote a shift towards self-care. Examples include common hurdles such as account registration and logging-in / authenticating within the app. Customer support has noted that many of the issues can be commonly resolved by keeping informed of how the device works and even quickly “Googling” the issue on the spot when unsure – suggesting that a majority of common issues can be handled properly with the right set of internal knowledge-sharing articles, public information, and proper guidance / access to troubleshooting guides from the device OEM.

2.3.2. Using app-level and log data

Even with a shift towards self-care, MVPD's may not be able to completely abdicate responsibility for remote troubleshooting during customer care calls. Particularly not when the MVPD app sitting on a

COAM device becomes the primary screen. Therefore, MVPD's should ensure that basic app level data can be collected and exposed real-time to customer care tools. This raises the stakes on care tool modernization, particularly a move towards a web-based approach and restful API's on the back end:

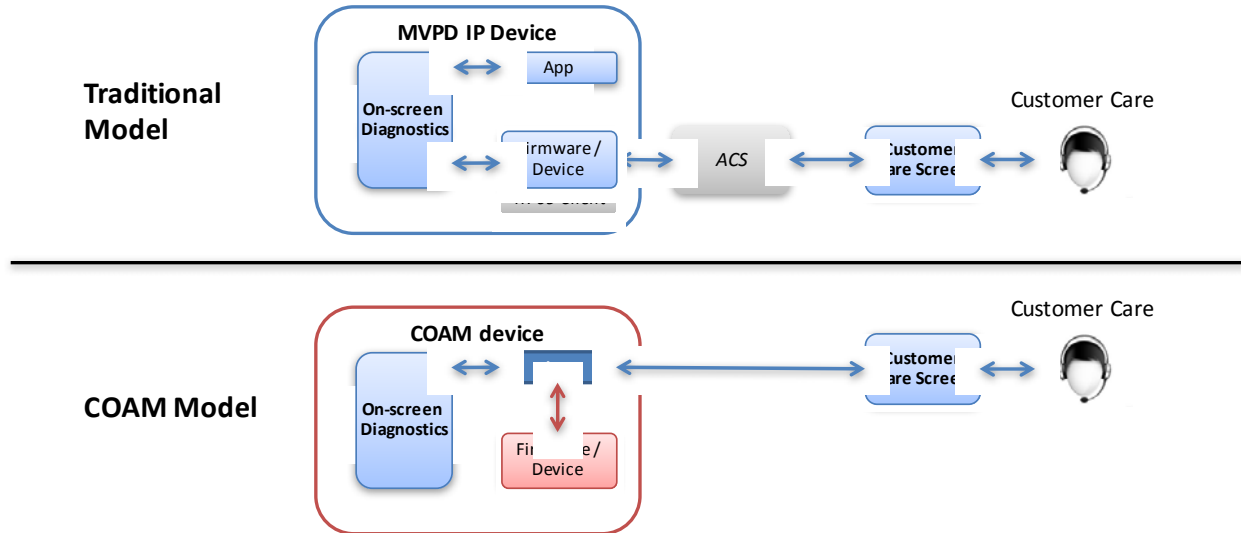


Figure 15 - In-app Diagnostics

App-level data will also have benefits for higher tiers of operations and development organization within an MVPD. Current operations rely heavily on polling mass populations of devices to understand key metrics such as non-responder rates. Historically this has been achieved through SNMP, which is impossible with COAM devices. Instead, all tiers of MVPD operations should rely on COAM app analytics, which raises the stakes for both app instrumentation and data analytics capabilities.

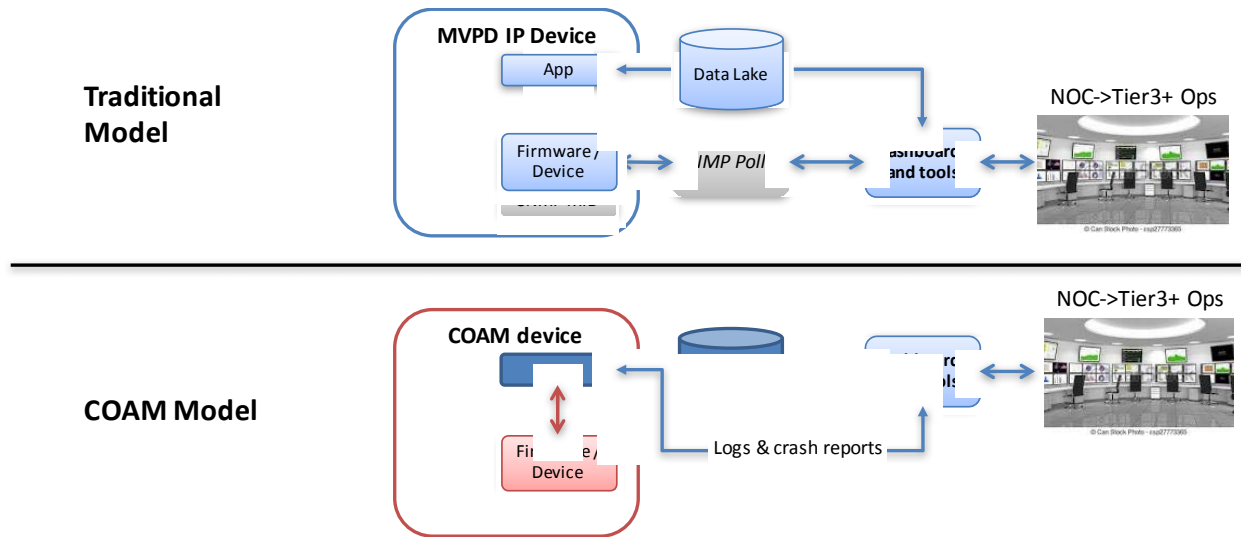


Figure 16 - COAM Population Health Monitoring

Similarly, MVPD’s must work with their COAM partners to get as much access as possible to logs and crash reports for root cause analysis when bugs do appear in production. Where available this must become a hardened part of operational root cause analysis for any MVPD who wishes to seriously explore COAM as a viable primary screen.

NOTE: This can be a point of contention with some COAM device providers who often do not allow access to such information unless a device is set to ‘developer mode’. In this scenario, MVPD’s can and should be a collaborative discussion point with COAM providers who may be able to instead share information offline rather than grant an MVPD direct access to log information at the firmware or device level.

2.3.3. In-app Wi-Fi diagnostics

COAM will inevitably shift the support model from video support to home networking and Wi-Fi, as more customer-owned devices enter the home and rely on strong connectivity to stream video. This is a departure from an MVPD connectivity model using MoCA or other wired means and therefore raises the stakes on Wi-Fi diagnostics. And those diagnostics will rely on app development and open-ness of COAM SDK’s to provide that information to on-screen or remote troubleshooting.

To examine this potential, we assessed the viability of in-app Wi-Fi diagnostics available through Roku’s SDK, which exposes a component interface that can grab device information, including that specific to its connection with a router or gateway. An MVPD can, therefore, leverage this functionality within its TV App. When fragments or packets get dropped, the MVPD app could then point to this troubleshooting console where the user is provided an interface with which she could self-diagnose and troubleshoot any connection issues that may be a root cause of an issue. The `ifDeviceInfo` interface for instance can support a number of methods that return values such as connection type (i.e. whether the device is connected by Wi-Fi or wired), the IP Address assigned, whether HDMI is connected etc. Device-level diagnostics data is thus handled at the app-level where the user is given a friendly UI and

clear instructions walking her through her own issue diagnosis, rather than a somewhat secretive OS-level menu that tends to be reserved for Roku's tech-savvy users. This can extend to other settings like bit-rate and pixel resolution adjustments; however, cautious in-app messaging and care agent guidance is highly recommended to prevent users from breaking their specific device configurations.

This troubleshooting console could also theoretically include a "speed-test" feature. By pinging a nearby server within the MVPD's network, this feature could return the user's download and upload speeds – effective in cases where a user is experiencing lagging issues in playback. For devices connecting via Wi-Fi, the app could run sequential tests while walking a user through locating the optimal location within the home for the fastest connection. As part of the MVPD's support plan, care agents could guide customers through this app and help educate with regards to the ideal speed for streaming. Speed-testing is also a key tool when it comes to upselling a customer. It provides a visual demonstration of their connection speed and in some cases, can lead them to conclude they need a faster connection if it's not meeting their streaming requirements.

- References: <https://sdkdocs.roku.com/display/sdkdoc/>

2.3.4. Provisioning and authentication considerations

In the traditional service provider model, one of the largest drivers of care calls are provisioning issues (e.g. misalignment of device to account association). Care tools, often tightly integrated with billing, send "hits" to STBs in order to correct this alignment in real time and fix issues for the customer.

With no known MAC address on COAM devices, MVPD's may encounter significant operational challenges related to provisioning if moving to COAM as the primary screen. Most MVPD COAM second screen implementations today rely on a username and password approach. It would behoove operators to begin moving towards a device to account registration model for every individual device on account. The registration process begins with the device sending a secure mechanism to the MVPD, the MVPD then generates the unique device ID that is associated with that device. In this way, anytime that device accesses the app, the MVPD does a simple lookup to know if the device is on the account. Even though auto-login is enabled in most MVPD TV app implementations, having the device registered should further improve convenience and user experience (i.e. without having to login again if token expires).

This approach is not still fool-proof. One of the biggest challenges is that it is difficult to generate a device ID that persists in perpetuity. Specifically, iOS and Android can often drive a need to re-register when upgrading OS versions since the device ID is often tied to the OS rather than the physical device itself – again the MAC address if obfuscated. It is therefore, operationally imperative that MVPD's drive this responsibility back to the end user via self-care tools – for example, having the TV app proactively alert the user that re-registration may be required once an OS upgrade is detected can prevent a spike in call volume. Messaging users proactively outside the app (e.g. through self-care portals/apps) can also safeguard against this re-registration.

2.3.5. Higher stakes for theft of service

Using an account to device registration model can also add another layer of security against theft of service. The username/password approach can prevent concurrent streams from occurring in an account at one time. But it does not prevent password sharing, an emerging revenue erosion threat should MVPD's move towards a COAM model as the primary screen.

Yet another layer of security that can and should be implemented by MSO's specifically is a feature called Home-based authentication (HBA). This feature will identify when a pay-tv customer is connected to their modem or gateway and automatically sign them into participating MVPD and programmer websites/apps on devices in the home. This approach works particularly well for fixed in-home devices serving the 10 foot screen, such as Apple TV, Roku, Connected TV's and Gaming Consoles. The approach does not work, however, for non-MSO's (i.e. non-broadband providers) nor will it work if the customer subscribes to video as a standalone service without broadband. Additionally, given the prevalence of Chromecast and other types of emerging Wi-Fi dongles, it is even more essential to enable device registration, particularly for iOS and Android devices that are used for both in home and out of home consumption.

3. 3rd party COAM apps (per FCC mandate)

This section builds on the previous analysis, but with the added complexity of complying with the FCC mandate to allow 3rd party applications access to MVPD back end infrastructure

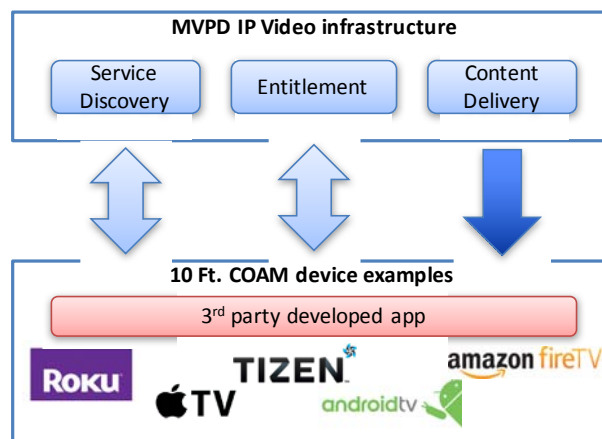


Figure 17 - 3rd Party COAM Apps

3.1. Develop

In this scenario, the MVPD will be responsible for developing and maintaining an SDK rather than the apps themselves. This SDK would need to be based on the aforementioned attributes of Service Discovery, Entitlement and Content Delivery. This is the first place that player abstraction, ideally based on HTML5 video standards, should be applied.

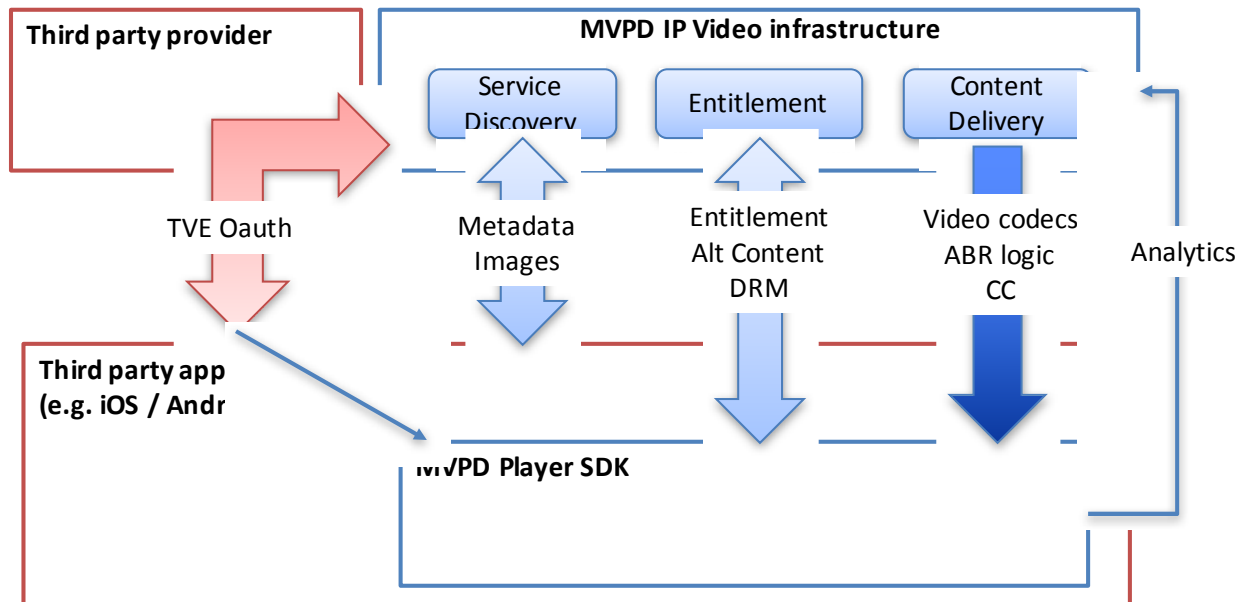


Figure 18 - Player Abstraction For Third Party Apps


The above illustrates what the FCC proposes is feasible in the foreseeable future from a technology perspective:

- A third party develops an app using an MVPD’s defined player SDK
- The third party app uses a three legged Oauth mechanism similar to TV Everywhere implementation in order to pass mapped credentials to the MVPD
- Service discovery API’s including metadata and images are exposed at the app level so that the third party can build their own navigation experience using MVPD data
 - o MVPD’s must ensure the right level of security for these API’s that would be publishable / accessible in the public domain
- The MVPD Oauth token is used by the abstracted player within the third party app which:
 - o Validates service and entitlement levels
 - o Determine the content that should be played including alternate content rules (i.e. blackouts and dynamic ad insertion). This is also where geolocation restrictions such as the in-home check must occur.
 - o Requests and receives DRM keys to decrypt MVPD content – NOTE: this is an area where using HTML5/Encrypted Media Extensions can help avoid DRM fragmentation
- Playout of video is enabled by video codecs and ABR logic resident in the player. Closed captioning is also enabled if selected by the user.

3.2. Deploy

At this point, it is essential to re-visit the previous deployment comparison to appreciate the implications of the FCC model on the MVPD:

Table 2 - Deployment Model per FCC Mandate

	End-to-end testing	MVPD Certification	App Certification & Publishing	Firmware / OS Update Schedule
MVPD guide on the MVPD STB	MVPD	N/A	MVPD	MVPD
COAM app	MVPD	N/A	3 rd party app ecosystem	COAM device 3 rd party
3 rd party app	3 rd party app ecosystem	MVPD	3 rd party app ecosystem	COAM device 3 rd party
MVPD Impact	N/A		N/A	N/A

Where the 3rd party app scenario becomes more complex is within the deployment model. As aforementioned, managing a secure and repeatable set of player APIs that allow 3rd party apps to authenticate subscribers, access appropriate content streams, metadata, etc. is both a logical and feasible starting point for addressing the mandate. However, it's paramount to qualify that with subscriber use of a 3rd party app, the 3rd party will 'own' the viewing experience. However, the MVPD will still own the **subscriber relationship**, as well as, being held ultimately accountable for the **quality** of the viewer experience.

With that in mind, this analysis posits that MVPDs must find the most appropriate (and cost effective) ways to maintain their desired degree of control and approval for any 3rd App presenting a different user experience with their content. This additional consideration should include tightly standardized testing approach & approval, as well as a negotiated SLA so that the MPVD can ensure desired levels of delivered video quality. Dictating participation in a standardized automated testing suite would ensure that level of quality while also not requiring a team of testers to consistently validate 3rd party app functionality with the MVPD IP streams.

It is also operationally essential that the MVPD have the ability to certify (or deny) app iterations from third parties. This is a complex undertaking that would require MVPD's to adjust the way they operate and further drive the need a 3rd party certification organization group to specifically address this need. Each third party would also need to be on-boarded beforehand, further driving complexity into MVPD business operations.

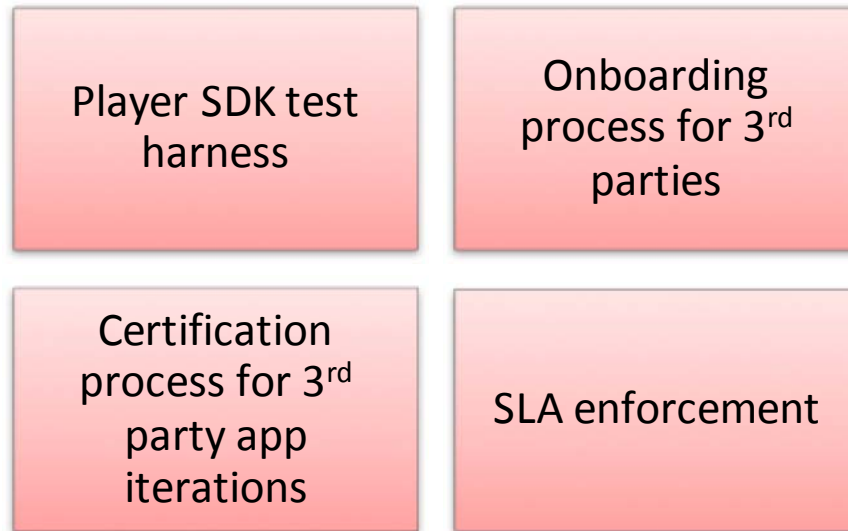


Figure 19 - Summary of Deployment Challenges with FCC Mandate

3.3. Operate

Finally, assuming that all of the aforementioned deployment requirements are met, the MVPD will need to operate 3rd party applications in a manner which is consistent with the quality of its current video service. Some requirements such as device registration and theft of service are possible to address via the player SDK. Others, however, are impossible to address when ceding ownership of the app to a third party. The issues lie most specifically with customer care:

- **Self-care:** Self-care interfaces and in-app Wi-Fi diagnostics would be unavailable to implement for end users
- **Traditional care:** App and log data would be more difficult to access for remote troubleshooting by customer care agents

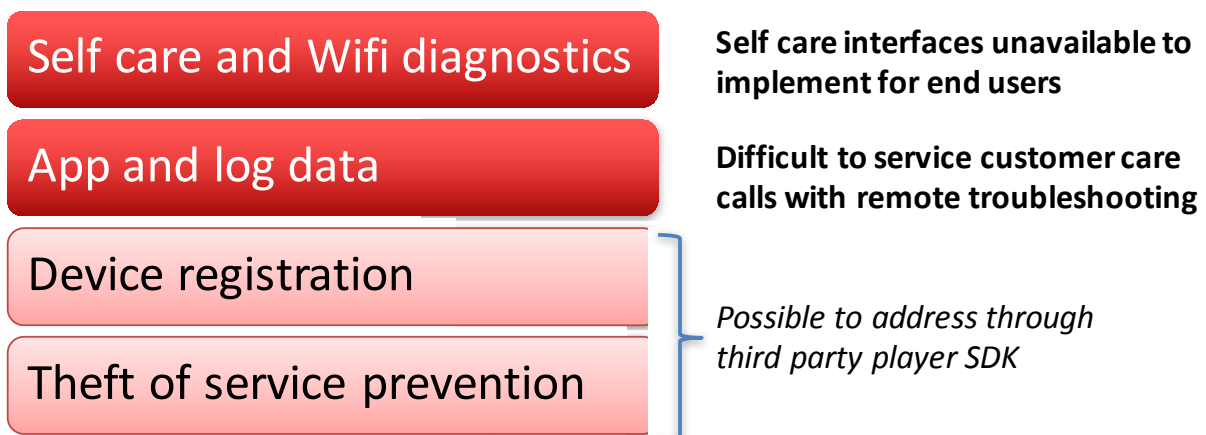


Figure 20 - Summary of Operational Challenges with FCC Mandate

Conclusion

This analysis concludes that developing MVPD apps for COAM devices as a primary screen alternative to leased STB's is a desirable outcome for the industry. To summarize the key learnings that should be considered to prepare MVPD's for a COAM future:

- Strive for modularity on both the front and back end of development
 - o Push for an open standard with HTML5
 - o Abstract the player, ideally through HTML5, but in a way that prepares for both MVPD apps or alternatively 3rd parties if mandated by FCC
 - o Implement a common abstraction layer where API's can be leveraged across the COAM app portfolio; considering a micro-services architecture and dynamic scripting for flexibility and scale
- Beef up testing capabilities in preparation for supporting a large portfolio of COAM devices in parallel to leased
- Drive COAM users towards self-care – but realize that traditional care must be supported and enabled through robust app instrumentation and integration with care tools

Conversely, while there are some technical possibilities to address the FCC's unlock the box mandate, the findings of this analysis suggest that the deployment and operational challenges make the outcome of 3rd party devices accessing MPVD video service extremely difficult. It is recommended that MVPD's use the findings of this paper and other similar analyses to demonstrate to the FCC that using MVPD COAM applications are a far more operationally feasible alternative that can also meet the stated need of offering consumers sufficient choice of platform.

Table 3 - Summary Comparison of MVPD v. 3rd Party Apps

	Develop	Deploy	Operate
MVPD apps	Drive for modularity <ul style="list-style-type: none"> • HTML5 standards • Player abstraction • Back end abstraction with microservices and dynamic scripting 	<ul style="list-style-type: none"> • Test team and environment robustness • Communication / SLA for firmware releases 	<ul style="list-style-type: none"> • Self care focus • In app Wifi diagnostics • App instrumentation and log / CR access • Device registration • Protect theft of service
3rd party	<ul style="list-style-type: none"> • 3rd party player SDK 	<ul style="list-style-type: none"> • <i>Standardized test validation suite</i> • <i>MVPD certification</i> • <i>3rd party onboarding</i> • <i>SLA enforcement</i> 	<ul style="list-style-type: none"> • <i>Self care focus</i> • <i>In app Wifi diagnostics</i> • <i>App instrumentation and log / CR access</i> • Device registration • Protect theft of service

Abbreviations

App	Application used for navigation and consumption of video
COAM	Customer Owned and Managed Devices
STB	Set-top box
MVPD	Multi-channel Video Programming Distributor

Bibliography & References

Appendix

Micro-services Overview

Software built as micro-services can, by definition, be broken down into multiple component services. Unlike traditional SOA or modular architectures that still retain the notion of monolithic back end applications, each micro-service can be deployed, tweaked, and then redeployed independently without compromising the integrity of an application. There is no dependency on an underlying application server or application. As a result, you only need to change one or more distinct services instead of having to redeploy entire applications. In a COAM world where scale, speed and control across your portfolio is critical, such an architecture is ideal.

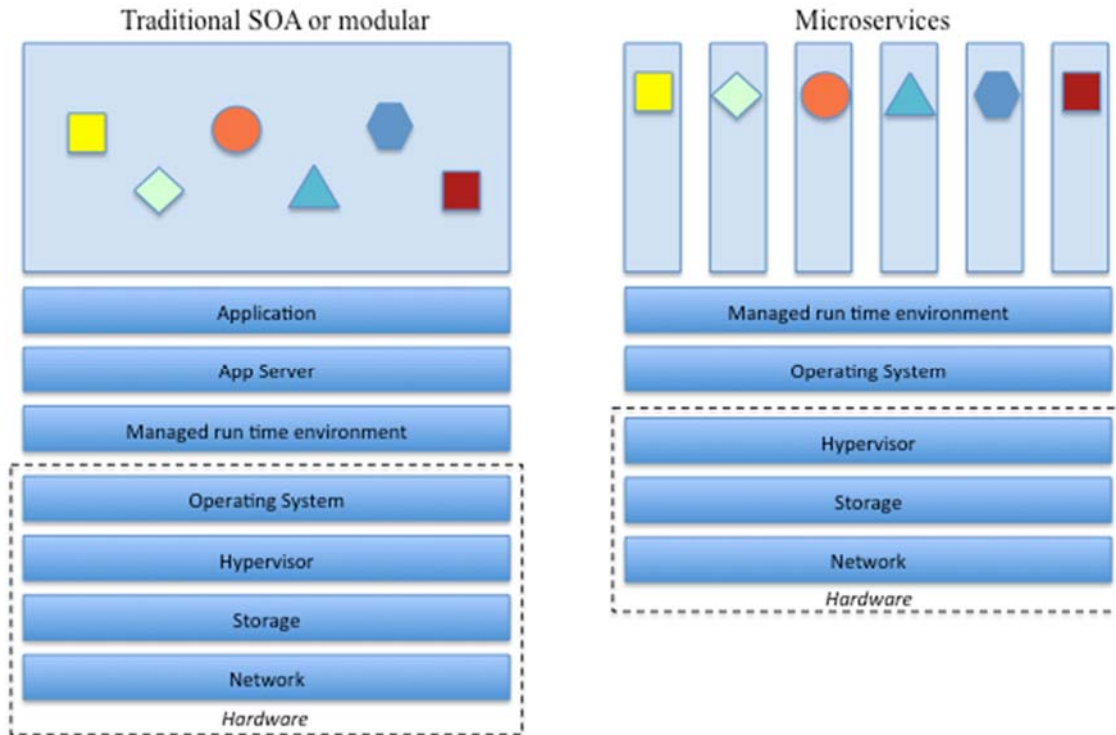


Figure 21 - Traditional SOA vs. Micro-services

Micro-services implications

Micro-services is one example of an architecture to regain control as the number of screens proliferates. The key for MVPD's is to establish an architecture that allows them to be nimble and responsive to an ever changing portfolio of devices, ecosystems and bespoke applications. In addition, micro-services minimize operational costs by reducing build and QA time, reducing overall effort across the deployment cycle as well as reduction in system maintenance activities.

Despite the benefits, a micro-services architecture can introduce some significant change for an organization. For example, when the number of back-end services increases, integration and managing whole products can become complicated. Further, partitioning the architecture into modular components services may require re-aligning developers who were historically linked to a specific application. Structuring your organization to support a broader micro-services portfolio becomes essential, but initially this can have an impact to current processes for troubleshooting, support and problem resolution.