ATSC 3.0 Datacasting with Core Networks

Luke Fay and Graham Clift (Sony Electronics, Inc.)

***Abstract*— Today's consumers are used to accessing data on their own time with protected one-to-one connections. That data can be personal or public video, software updates, navigation aid, files in general, etc. For sending data out to many devices, multicasting via a broadcast signal that is inter-connected in the cloud (e.g., a core network) provides advantages of time and resources. ATSC 3.0 offers datacasting via broadcast television signals with advantages of built-in file repair as described in A/331[1].**

***Index Terms*—ATSC 3.0, NEXTGEN TV, datacasting, core networks, Sony Xperia, digital terrestrial broadcasting.**

I. Introduction

Datacasting with Over the Air (OTA) broadcast signals can be available to public and private organizations. To all those interested in delivering information in the most efficient manner, ATSC 3.0 has enabled datacasting possibilities to a variety of devices, even memory constrained platforms. The question becomes, "How much data do you want to send, and how fast do you want it get there?" ATSC committees are working to develop a standard that links TV stations together nationwide. Whether broadcasters are privately owned or part of a station group, opening-up their spectrum for datacasting offers new business possibilities.

Having one webpage (URL) where customers can order delivery of their files securely to the public or to specific private devices provides convenient ease of use as a one-stop shop file delivery. EW Scripps, Nexstar and Hewlett Packard Enterprise (HPE) have joined forces to develop a Core Network that connects their stations to one online portal. From that online portal with a variety of login security measures, station engineers can configure Services and customers can also order delivery of their files. This online portal can onboard many stations nationwide and be privately secured to particular entities (i.e., Nexstar is the only one to access their stations).

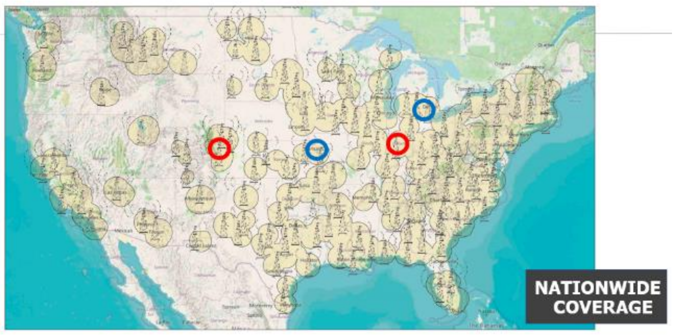


Figure I.1 National Coverage

National coverage with these broadcast station groups is shown in Figure I.1.

At the time of this writing, four Designated Market Areas (DMA) that have connected to the Core Network. EW Scripps stations in Kansas City and Detroit, and Nexstar stations in Denver and Indianapolis. This pilot program for starting

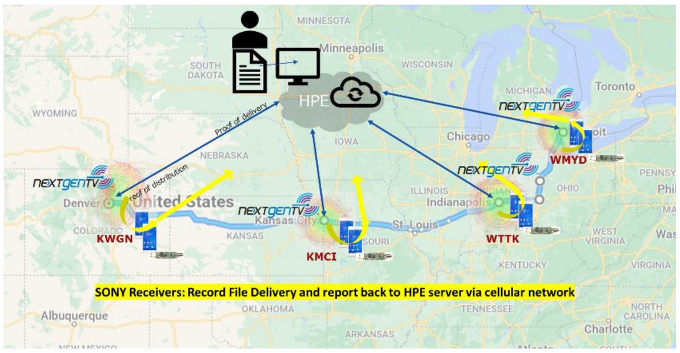
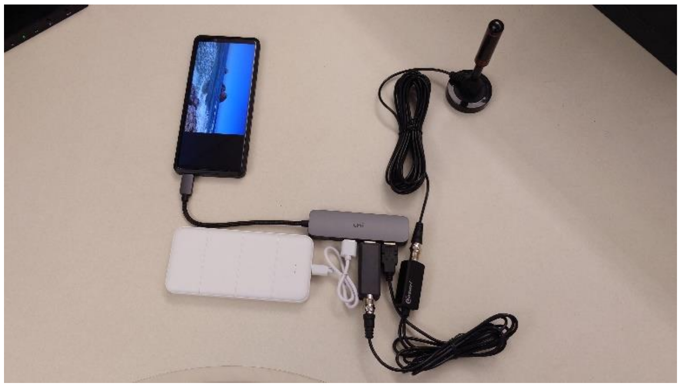


Figure I.2 GoldenSpike Coverage

operation was named 'Goldenspike'. Combined, Nexstar and EW Scripps can reach around 90% of US households. This pilot program for starting datacasting is shown in Figure I.2.

Sony Electronics provided the receivers (Figure I.3) to monitor all Services, linear Audio/Video programming and datacasting in general. Specifically, those receivers included a Sony Xperia phone, Sony USB based ATSC 3.0 dongle, UHF antenna, antenna amplifier, a USB hub to connect all those items plus a portable power pack to extend mobile test time. Sony Electronics also developed JAVA code to run on Android Xperia Phones with 5G / LTE radios; models Xperia Pro, Xperia 10, Xperia 10+, Xperia 5, and Xperia 1. Five phones were provided to each DMA so broadcasters can run their test plans. Reports of signal reception quality and other parameters are sent back to the Core Network using HPE defined API's. This monitor testing included GPS coordinates for broadcasters



to get real time

Figure I.3 Sony Receiver

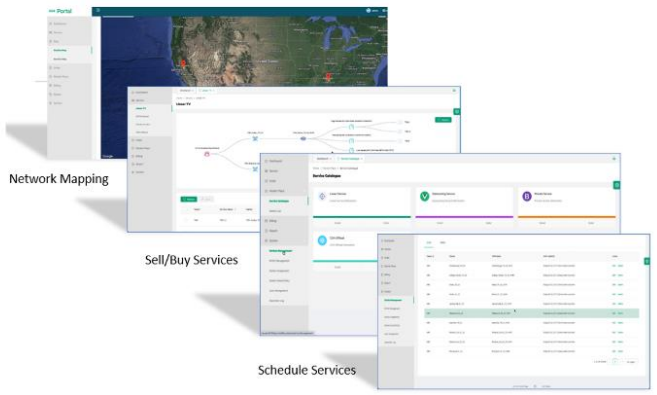
feedback of their signal quality across their markets. File size and reception time were also included in those reports so that delivery times between ordering a service on the Core Network and reception to all markets could be measured.

This paper will explain the test setup for the Core Network, Sony Electronics receiver configuration and results reporting back to the Core Network. Datacasting can have a few tiers of service, e.g., real-time feedback, once a week feedback, feedback when possible or no feedback at all. Cellular connections allow for real-time feedback, a highly desirable model, enabling accurate reads across markets to those customers requiring live feedback data.

II. Core Network

HPE developed the Core Network that EW Scripps and Nexstar use for this trial.

Figure II.1 Core Network Portal



HPE's Core Network portal (Figure II.1) enables orchestration of spectrum and payload allocations across multiple broadcast stations in many markets. ATSC 3.0 payload traffic is governed via a user-friendly Core Network 'portal' for broadcasters to manage. HPE's Core Network index page shows a diverse list of marketplace services to start.

There is also network mapping page to see which markets are available and how far each ATSC 3.0 station reaches, a schedule page to coordinate timing of datacasting or any other service of interest, a sell/buy services page to order up services, a reports page to validate data delivery and receive signal quality feedback, billing, and other relevant pages. This paper will focus on the receiver reports from fielded devices, example shown in Figure II.2.

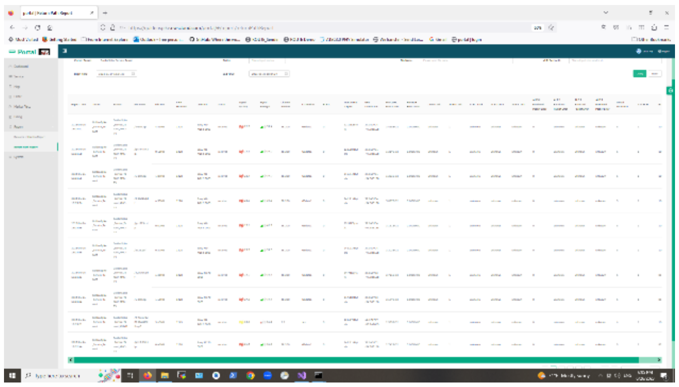


Figure II.2 Return Path Reports

The portal entry URL page allows broadcasters to configure their services per their business use-cases. Currently, test stations use two QoS settings, stationary and mobile running simultaneously. Stationary services can operate around 20dB SNR and mobile can operate around 5dB, but there are more than one setting for each Service depending on how much payload is desired. The datacasting payload is put on the mobile QoS payload and continually repeats. Sony receivers report back to the Core Network upon successful reception of each file, the smaller the file, the faster the reports. This is useful if RF reception quality is of interest to broadcasters who want to see in real time the reach of their signals.

Datacasting service is delivery of non-real time files, which is signaled as an Application-Based Service. This is Service Category = 3 in A/331[1] Table 6.4. There is another Service Category for DRM Data Service (DRM Data), but these files are not encrypted, and receiver code was already developed to look for applications to store files. For fielded devices that render APP-based service, the @hidden flag in the SLT can be set and only Sony receivers will know to look for this App-Based Service on a specific UDP address and port number.

III. Ordering Datacasting service

Start a new order for datacasting under Market Place tab in the Core Network's Service Catalog (Figure III.1). There are many different service types, datacasting is one of them.

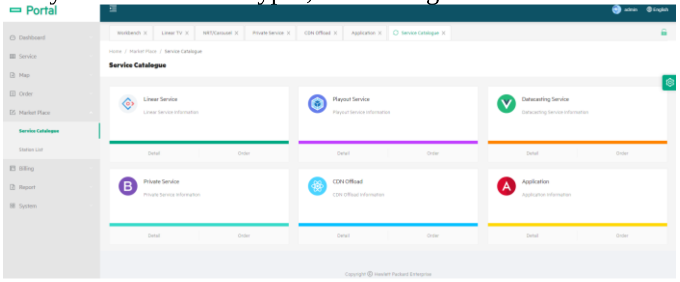


Figure III.1 Service Catalog

Under Datacasting Service, there is 'Detail' for listing current campaigns and 'Order' for placing a new datacasting campaign. NRT Service Order allows a customer to select a Service name, maximum bit rate for delivery, market area selections (which cities and/or multiple market areas can be selected if needed), which quality of service (high payload or low), how many times to carousel the file out for delivery among other details (Figure III.2).

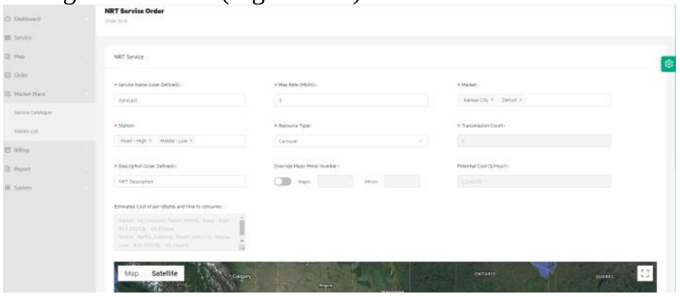


Figure III.2 NRT Service Order

Once the service is ordered, a file can be selected to upload to the Core Network under the Service tab's NRT/Carousel menu item. The Core Network can also fetch files via integration with S3 Buckets and ABS Blobs or via SFTP or HTTPs. After file is uploaded or fetched, a list of datacasting service status is shown in that Service NRT/Carousel tab (Figure III.3).

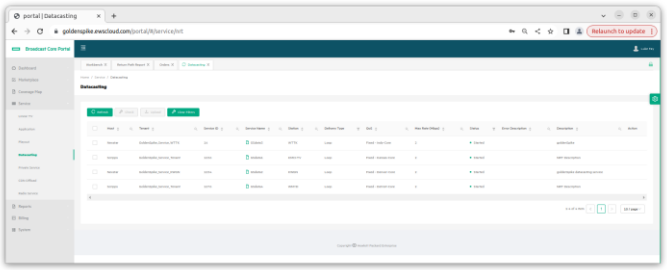


Figure III.3 Datacasting List

To verify which datacasting services are operational, click on the Service Name (e.g., GSdata3) and a window will pop-up and display the current datacasting status (Figure III.4).

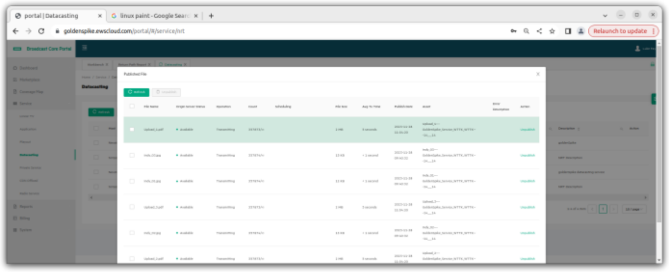


Figure III.4 Datacasting Status

IV. Receiver Platform Considerations

Datacasting receivers can range from small mobile sensors to high end CPU machines collecting information. Those small mobile or stationary devices have limited resources. While the Sony phones have ample memory, they are not likely to be used by customers looking to update kiosks, billboards, or sensors. EW Scripps put on a call for some Reference Design Kits (RDK) which resulted in contributions using Raspberry Pi's. An example list of devices and their capabilities is provided in Table IV.1.

Table IV.1 Device Capabilities

Platform	CPU	Memory	OS
Raspberry Pi2 Model B	900MHz Quad-core ARM Cortex-A7	1GByte RAM	Raspberry Pi OS (Raspbian)
Raspberry Pi3 Model B	1.2GHz Broadcom BCM2837, Quad-core ARM Cortex-A53	1GByte RAM	Raspberry Pi OS (Raspbian)
Raspberry Pi4 Model B	1.8GHz Broadcom BCM2711, Quad-core ARM Cortex-A72	Up to 8GByte RAM	Raspberry Pi OS (Raspbian)
Geniatech XPI-S905x3 Single Board Computer	1.8GHz Amlogic S905X3 Quad-core ARM Cortex-A55	2GByte RAM, 8GByte eMMC FLASH	Android 9.0 OS
Geniatech DB982 ATSC 3.0 TV Motherboard	1.8GHz Amlogic T982 Quad-core ARM Cortex-A55	2GByte RAM, 16GByte eMMC FLASH	Android 11.0 OS
A95X F3 TV-Box	1.9GHz Amlogic S905X3 Quad-core ARM Cortex-A55	4GByte RAM, 64Gbyte eMMC FLASH	Android 9.0 OS
Nvidia Shield set-top box	1.9GHz Nvidia Tegra X1+ (8-core based ARM Cortex-A57)	3GByte RAM, 16GByte eMMC FLASH	Android 11.0 OS
Qualcomm SM8250 Snapdragon 865	Octa-core (1 x 2.84 GHz Cortex-A77 & 3 x 2.42 GHz Cortex-A77 & 4 x 1.80 GHz Cortex-A55)	12GB RAM, 512GBYte eMMC FLASH	Android 10.0 OS

Note the Memory column which indicates how much is available to the OS and all software programs for processing of data. The latest Raspberry Pi OS with desktop in 32-bit version is 1,239Mbyte in size, with a recommended 2GByte RAM. To put that into perspective, other OS sizes and RAM recommendations are provided in Table IV.2.

Table IV.2 OS Memory Requirements

Operating System	Minimum RAM Requirements	Recommend RAM
Microsoft© Windows©10 64-bit	2GB	8GB
Microsoft© Windows©10 32-bit	1GB	4GB
Microsoft© Windows©8 64-bit	2GB	8GB
Microsoft© Windows©8 32-bit	1GB	4GB
Microsoft© Windows©7 64-bit	2GB	8GB
Microsoft© Windows©7 32-bit	1GB	4GB
Mac© OS X Sierra	4GB	8GB
Mac© OS X El Capitan	2GB	8GB
Mac© OS X Yosemite	2GB	8GB
Mac© OS X El Mavericks	2GB	8GB
Mac© OS X High Sierra	2GB	8GB
Mac© OS X Mojave	2GB	8GB
Linux	8MB	16MB
Raspbian with Desktop	450MB	2GB
Raspbian Pi OS Lite (32-bit)	256MB	
Android 10 (32-bit)	896MB	
Android 10 (64-bit)	1280MB	

Sources: https://www.crucial.com/articles/about-memory/how-much-ram-does-my-computer-need , https://www.oreilly.com/library/view/running-linux-third/156592469X/ch01s09.html , https://www.tomshardware.com/news/raspberry-pi-4-how-much-ram, https://source.android.com/docs/compatibility/10/android-10-cdd

That fills the available RAM space already, but adding a few programs increase the RAM size recommendations. Table IV.3 shows a few programs' RAM minimum requirements.

Table IV.3 Program Memory Requirements

Software	Minimum RAM Requirements	Recommend RAM
Adobe© InDesign©	2GB	16GB
Adobe© Illustrator©	1GB	16GB
Adobe© Photoshop©	2GB	32GB
Adobe© Premiere© Pro	8GB	32GB
Adobe© After Effects©	4GB	64GB
GIMP Image Editor	75MB	
Chromium with 59Tabs open	2.3GB	
OpenShot Video Editor	120MB	
LibreOffice Calc	75MB	
Scratch 3 Desktop	400MB	

Sources: https://www.crucial.com/articles/about-memory/how-much-ram-does-my-computer-need , https://www.tomshardware.com/news/raspberry-pi-4-how-much-ram

Memory constraints of 2GByte of RAM already hinders operation of some Operating Systems. With customers loading their own software on these devices to perform datacasting, there is likely only a very limited amount of RAM for processing incoming files. However, these memory constrained platforms can still supply large files (>1GByte) to an SD Card or transfer to an external hard drive.

Delivering large files to memory constrained platforms carries a risk of dropping packets over time. To help protect against dropped packets and having to wait for files to carousel around again for the receiver to pick up lost packets, ATSC 3.0 enables delivery protection with a fountain code (Raptor-Q). This is specified in RFC 5053 which is referenced in ATSC A/331[1]. For each file delivered, there are extra bits provided incase repair of the file is needed. In this study, 10% repair was used for each file.

Each Sony Xperia phone connected to an ATSC 3.0 USB dongle along with portable power pack and antenna.

These were all put into a case for easy transporting, picture on the right. Xperia Pro phones come with Android 10 OS running on Qualcomm SM8250 Snapdragon 865 5G chipsets which have Octa-core (1x2.84 GHz Cortex-A77 & 3x2.42 GHz Cortex-A77 & 4x1.80 GHz Cortex-A55) CPU with Adreno 650 GPU. There is 12GByte of RAM available for all applications in the phone, and Sony's ATSC 3.0 application is just one of them but is expected to work with large files.

V. ATSC A/331 Standard

Accommodating these small memory platforms required an amendment to ATSC A/331[1]. Existing ATSC A/331 standard already includes support for Raptor-Q, a fountain-based code that sends repair bits to fix missing data in packets. These repair bits are sent at the end of a file using Repair Flow signaling. This model assumes the receivers can store the entire file in RAM and only perform repair at the end of file delivery. Many receivers do not have the kind of memory for large files, especially files > 1GByte. ROUTE operation with Source Flow signaling already breaks a file into a collection of Source Blocks for sending the files in pieces.

These pieces are delivered in order with this Source Flow and total file size is signaled in the S-TSID with @ContentLength attribute. Receivers allocate enough memory for the entire file and then collect the file pieces (Source Blocks) until entire file is collected. Example is tune-in typically occurs in the middle of a file and Source Block (e.g., #G) is collected, followed by Source Blocks (e.g., #H, #I) etc. until the file carousel wraps and Source Block (e.g., #A, #B) beginning Source Blocks are collected up to Source Block (e.g., #F). When receiver collects all source blocks adding to @ContentLength, the file is completely received.

Sony Electronics contributed to the idea of breaking up the files into small partitions that contain a number Source Blocks, just large enough to fit into 1/3 of a devices' memory and adding MD5 checksums signaled in the Repair Flow for each partition. This enables receivers to verify MD5 checksum calculations to declare file is correctly received. The number 1/3 comes from the Raptor-Q algorithm, where intermediate symbols are generated from the original Source Blocks in 1:1 fashion and then memory to perform the repair.

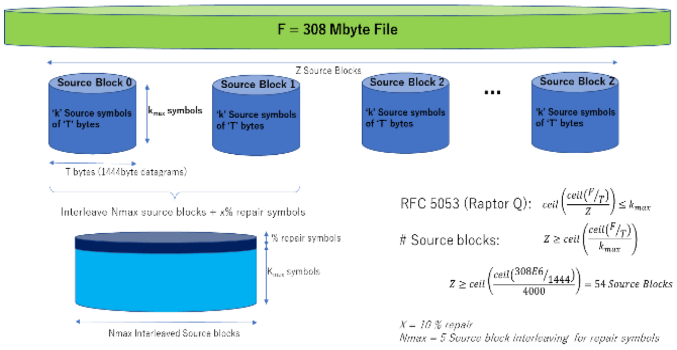

F = 308 Mbyte File
Z Source Blocks
Source Block 0
Source Block 1
Source Block 2
Source Block Z
'k' Source symbols of 'T' bytes
k_{max} symbols
T bytes (1444byte datagrams)
Interleave Nmax source blocks + x% repair symbols
% repair symbols
K_{max} symbols
Nmax Interleaved Source blocks
RFC 5053 (Raptor Q): $ceil\left(\frac{ceil(F/T)}{Z}\right) \le k_{max}$
# Source blocks: $Z \ge ceil\left(\frac{ceil(F/T)}{k_{max}}\right)$
$Z \ge ceil\left(\frac{ceil(308E6/1444)}{4000}\right) = 54\ Source\ Blocks$
X = 10 % repair
Nmax = 5 Source block interleaving for repair symbols

Figure V.1 RFC 5053 Equations

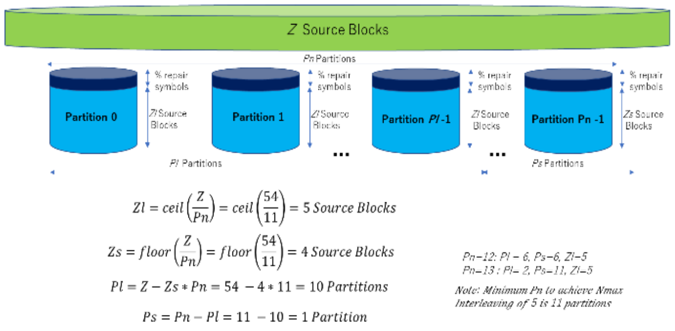

Z Source Blocks
Pn Partitions
Partition 0
Partition 1
Partition Pl -1
Partition Pn -1
% repair symbols
Zl Source Blocks
Zs Source Blocks
Pl Partitions
Ps Partitions
$Zl = ceil\left(\frac{Z}{Pn}\right) = ceil\left(\frac{54}{11}\right) = 5\ Source\ Blocks$
$Zs = floor\left(\frac{Z}{Pn}\right) = floor\left(\frac{54}{11}\right) = 4\ Source\ Blocks$
$Pl = Z - Zs * Pn = 54 - 4 * 11 = 10\ Partitions$
$Ps = Pn - Pl = 11 - 10 = 1\ Partition$
Pn=12: Pl = 6, Ps=6, Zl=5
Pn=13 : Pl= 2, Ps=11, Zl=5
Note: Minimum Pn to achieve Nmax Interleaving of 5 is 11 partitions

Figure V.2 Repair Flow Extensions

Figure V.1 shows a 308Mbyte file split into a number of Source Blocks [#0…Z]. For 10% repair, the example shows Z equals 54 Source Blocks. Only after those 54 Source Blocks are sent is the repair bits delivered over-the-air. Figure V.2 shows the collection of Source Blocks into a partition where repair bits are sent at the end of a partition. The example here shows 11 partitions in total, meaning repair is parsed in time across 11 partitions. This has the added benefit of tolerating long packet dropouts. The equations need a few input parameters. Those parameters are added to A/331[1] Section A.3.3.2.3.1 ATSC Extensions to the **`FDT-Instance.File Element`** and Section A.4.3.2 Semantics for Repair Flow. The edits are highlighted later.

Further enhancements are made by sending packets out of order, namely interleaving payload across Source Blocks within the Partition. This spreads out missing packets in time and allows for longer dropouts in packets. How much repair is needed? It depends on the environment and length of time a broadcaster would like to cover. In previous field studies, it was found that even with robust PLP operation there is packet loss. Traveling behind buildings, in a tunnel, at the bottom of hills, etc. packet loss will occur no matter what the operating point (QoS). Sending repair should be thought of how long to cover packet loss, not tied to any QoS operating point. Interleaving across Source Blocks increases the amount tolerable, an example for 10% repair across a few modulations is shown in Table V.1. Repair flow includes signaling for Application Layer – Forward Error Correction (AL-FEC) which uses the Raptor-Q fountain code.

Adding the AL-FEC with out-of-order packet delivery increases packet loss dropout coverage from 0.2msec to over 6 second coverage. This is more practical number and can be increased with more repair bits, but there is a tradeoff. More repair bits add more data sent over the air, which cuts into source flow payload that delivers the desired data. These tradeoffs of %repair vs channel conditions could be future study, but hopefully the Table V.1 gives a glimpse into what is possible with AL-FEC.

This extended protection against packet loss is an advantage of Repair Flow. The needed parameters are:

1. Amount of cache memory to perform repair
2. Out of order delivery signaling
3. Percentage of repair
4. CRC32 checksum value to validate repair operation for each partition.

Table V.1 Repair Flow Advantage

	PHY					AL-FEC						
	Cells* (Useful Carriers)	Time [us]	Cell Rate [Mcells/s]	Average TI length [s]	CI depth (CI) [s]	Packet size (T) [bytes]	QPSK Bitrate [Mbps]	Packet (T) period [msec/packet]	Source Block size (k_{max}) [packets]	Max Partition size [bytes]	% Repair	Estimate Dropout tolerance [s]
QPSK	6144	1194.67	5.1428571	0.2	0.4	1444	3.39322533	0.003404431	4000	20000	10	6.808861111
16QAM	6390	1194.67	5.3485714	0.1	0.2	1444	3.52895434	0.003273491	4000	20000	10	6.546981838
64QAM	13435	2389.33	5.6228571	0.1	0.2	1444	3.70992636	0.003113808	4000	20000	10	6.22761687
256QAM	27853	4778.67	5.8285714	0.075	0.15	1444	3.84565538	0.003003909	4000	20000	10	6.007818627
1kQAM	27853	4778.67	5.8285714	0.075	0.15	1444	3.84565538	0.003003909	4000	20000	10	6.007818627
4kQAM	27853	4778.67	5.8285714	0.075	0.15	1444	3.84565538	0.003003909	4000	20000	10	6.007818627

These are signaled in A/331[1] Annex A, tables A.3.4 and table A.4.1. That text is copied here for the reader's convenience.

Informative Table V.2 lists the ATSC-defined extensions to the **`FDT-Instance.File`** element. The semantics of the additional attributes described in Table V.2 are given below the table. The additions for operating with memory constrained platforms are highlighted.

Table V.2 FDT File Element

Element or Attribute Name	Use	Data Type	Description
@appContextIdList	0..1	afdt:uriList	A space-separated list of URIs representing one or more unique Application Context Identifiers.
@filterCodes	0..1	afdt:listOfUnsignedInt	A space-separated list of 32-bit unsigned integers representing Filter Codes that apply to this file.
@maxCacheMemory	0..1	unsignedint	The maximum memory required at any time by the receiver to cache the data when the packets are being delivered out-of-order.
@outOfOrder	0..1	Boolean	Flag indicating out-of-order delivery of source packets if value is "true" or in-order delivery of packets if value is "false".

@maxCacheMemory – When **`FDT-Instance.order`** flag is false, this optional 32-bit unsigned integer attribute represents the maximum memory required at any time to keep the received file data in cache. This allows the receiver to assess its capability to receive the data in non-persistent storage prior to block transfer of a completed block of contiguous data to persistent storage or for the receiver to repair a block of data in the case when repairFlow is utilized. When not present the **`FDT-Instance.maxTransportSize`**, the LCT Header Transfer Length or the Entity Mode Content-Length should be used by the receiver. If **`FDT-Instance.order`** is true, this attribute has no meaning. When Repair Flows are in use, the value of @maxCacheMemory shall be the summation of the source flow symbol payload bytes across the maximum concurrently interleaved source blocks.

@outOfOrder – This Boolean attribute shall indicate whether the payload of ROUTE packets carrying the objects, or a portion thereof, of the source flow are delivered out of the order of their generation. When the value is "true", arbitrary delivery is indicated. When the value is "false", in order of delivery is indicated. When this attribute is absent, the default value shall be "false".

The **`RepairFlow`** element is a child element of **`S TSID.RS.LS.`**

While the XML schema specifies the normative syntax of the **`RepairFlow`** element, informative Table V.3 describes the structure of the **`RepairFlow`** element in a more illustrative way.

The semantics of the elements and attributes of **`RepairFlow`** are given in Table V.3 with additions for memory constrained platforms highlighted. Semantics of the additional attributes follow the table.

Table V.3 Repair Flow Elements

Element or Attribute Name	Use	Data type	Description
RepairFlow		stsid:rprFlowType	Repair flow carried in the LCT channel.
FECParameters	0..1		FEC Parameters corresponding to the repair flow.
@maximumDelay	0..1	unsignedInt	Maximum delivery delay between any source packet in the source flow and the repair flow.
@overhead	0..1	unsignedShort (percentage)	FEC overhead in a ROUTE packet represented as a percentage.
@minBuffSize	0..1	unsignedInt	Required buffer size to handle all associated objects that are assigned to a super-object.
@fecOTI	1	hexBinary	FEC-related information associated with an object as well as FEC information associated with the encoding symbols of the object and is to be included within this declaration and applies to all repair packets with the repair flow.
@percentRepair	0..1	UnsignedShort	The maximum ratio of repair symbols to source symbols as a percentage
@ChecksumList	0..N	unsignedInts	List of CRC32 hexadecimal formatted checksums for each source block in the order of the source blocks.
ProtectedObject	0..N		List of the source flow(s) protected by this repair flow and the details on how the protection is done. It also defines how certain delivery objects of a collection of objects are included in the repair flow.
@sessionDescription	0..1	string	Specifies the session description information for the source flow.
@tsi	1	unsignedInt	The Transport Session Identifier (TSI) for the source flow to be protected.
SourceTOI	1..N		The Transport Object Identifier (TOI) value(s) of the delivery object(s) corresponding to the TOI of a given FEC (super-)object delivered by the repair flow.
@x	0..1	unsignedShort	Value of the constant X for use in deriving the TOI of the delivery object from the TOI of the FEC (super-)object.
@y	0..1	unsignedShort	Value of the constant Y for use in deriving the TOI of the delivery object from the TOI of the FEC (super-)object.

@percentRepair – An optional 16-bit unsigned integer that equates to the maximum ratio of repair symbols to source symbols as a percentage. Absence of this attribute means the AL-FEC repair data size is not provided.

@checksumList – An optional space separated list of 32bit unsigned integers in hexadecimal format that equates to the calculated CRC32 checksums of each source block. This list may be used by the receiver to determine the accuracy of received and/or repaired data in a source block. In its absence the integrity of the complete received and/or repaired file may be available through the optional FDT-Instance MD5 Checksum, however this CRC32 checksum list, when present, allows the receiver to more finely assess the need to replace data at a source block level of resolution.

VI. Software

A. *Hewlett Packard Enterprise API's*

HPE defined a few API's that represent the minimum requirements to interface with their Core Network. Sony Electronics added more data to those API's for monitoring signal conditions and phone status. This allows real time data feedback of useful data to HPE, Sony Electronics, broadcasters and datacasting customers. Results boiled down to three API's of 'Create', 'Update' and 'Delete' which are first defined, followed by usage descriptions and then an example.

Definition

URL: [Core Network URL]
Method: POST
[HEAD]:
authorization : Basic YWRtaW46cGFzc3dvcmQ=

[Body]

1. For Create

```
{
    "date":"2022-2-25T10:55:45Z",
    "operation":"create",
    "fileName":"AdvanceSearchSample---
Amazon_Fire_Chattanooga_TN_26---866---Amazon_Fire.txt",
    "size":"2M",
    "deviceID":"device002",
    "status":"received",
    "description":"description",
    "signalStrength":"signalStrength",
    "signalQuality":"signalQuality",
    "channelNumber":"channelNumber"
    "speed": "transmissionSpeed",
    "nrtServiceID": "channelNumber",
    "nrtPlpID": "PLP Identifier",
    "GPS coordinates": "gps",
    "Pre-LDPC Error Count": "plec",
    "Pre-BCH Error Count": "pbec",
    "ALP [i] Lock": "alp[i]Lock",
    "ALP [i] Baseband Packet Error": "alp[i]bpe",
    "Global Service ID": "gsId",
    "# of PLPs": "plps",
    "Battery Life": "batteryLife"
}
```

2. For Update

```
{
    "date":"2022-2-25T11:55:45Z",
    "operation":"update",
    "fileName":"AdvanceSearchSample---
Amazon_Fire_Chattanooga_TN_26---866---Amazon_Fire.txt",
    "size":"2M",
    "deviceID":"device002",
    "status":"finished",
    "description":"description",
    "signalStrength":"signalStrength",
    "signalQuality":"signalQuality",
    "channelNumber":"channelNumber"
    "speed": "transmissionSpeed",
    "nrtServiceID": "channelNumber",
    "nrtPlpID": "PLP Identifier",
    "GPS coordinates": "gps",
    "Pre-LDPC Error Count": "plec",
    "Pre-BCH Error Count": "pbec",
    "ALP [i] Lock": "alp[i]Lock",
    "ALP [i] Baseband Packet Error": "alp[i]bpe",
    "Global Service ID": "gsId",
    "# of PLPs": "plps",
    "Battery Life": "batteryLife"
}
```

3. For Delete

```
{
    "date":"2022-2-25T12:55:45Z",
    "operation":"delete",
    "fileName":"AdvanceSearchSample---
Amazon_Fire_Chattanooga_TN_26---866---Amazon_Fire.txt",
    "size":"2M",
    "deviceID":"device002",
    "status":"deleted",
    "description":"description",
    "signalStrength":"signalStrength",
    "signalQuality":"signalQuality",
    "channelNumber":"channelNumber"
    "speed": "transmissionSpeed",
    "nrtServiceID": "channelNumber",
    "nrtPlpID": "PLP Identifier",
    "GPS coordinates": "gps",
    "Pre-LDPC Error Count": "plec",
    "Pre-BCH Error Count": "pbec",
    "ALP [i] Lock": "alp[i]Lock",
    "ALP [i] Baseband Packet Error": "alp[i]bpe",
    "Global Service ID": "gsId",
    "# of PLPs": "plps",
    "Battery Life": "batteryLife"
}
```

Usage

################
A post is sent to the server in the following three cases:
1. When you receive a new file ("create")
2. When the contents of an existing file are changed ("update")
3. If the folder is full and old files are deleted ("delete")
################

The unit of signalStrength is dBm, and this value is the same as RSSI in the signal status menu.

The unit of signalQuality is dB, and this value is the same as Quality in the signal status menu.

The unit of 'speed' is Kbyte per second and represents transmission speed for which the first and last packet reception times are marked along with the file size of the NRT file to calculate the value.
################

Example

Request Body:

```
{
  "date": "2022-07-25T15:03:52Z",
  "operation": "create",
  "fileName": "PIC13_TEST---John_Dallas---192---John.jpg",
  "size": "3329661",
  "deviceID": "Sony XQ-AQ62 0013",
  "status": "received",
  "description": "description NG test",
  "signalStrength": "-52",
  "signalQuality": "183",
  "channelNumber": "6.2 TBN",
  "speed": "334133",
  "nrtServiceID": "6.1 DC1",
  "nrtPlpID": "0",
  "GPS coordinates": "39.723988,-104.98528",
  "Pre-LDPC Error Count": "2.546E-02",
  "Pre-BCH Error Count": "0.000E+00",
  "ALP 0 Lock": "1   ",
  "ALP 0 Baseband Packet Error": "0   ",
  "Global Service ID": "tag:trivenidigital.com,2019:atsc3\/us\/504\/15",
  "# of PLPs": "1",
  "Battery Life": "100"
}
```

B. GPS Coordinate

Validating broadcaster signal coverage requires location knowledge. Each Sony Xperia phone contains GPS radios / calculations which can be reported back to the Core network. This is one advantage to using the phones rather than a Raspberry PI device which would need an additional GPS module. Location can come from cellular network triangulation or the GPS satellite directly. Each phone here used the GPS satellites.

Sony's ATSC3.0 Application was able to request the phone's GPS coordinates and report back. Location coordinates are only reported upon correct file reception, so large files took longer to deliver hence reporting was more sparse for these larger files. Indianapolis has the smallest files of 10kByte and had many reports.

C. Cellular option

Connection to the Core Network is made with the cellular radio, which requires a SIM card and dataplan registration with the local Mobile Network Operator (MNO). Other radios like Bluetooth, WiFi, etc. are possible for connections to the Core Network but cellular option made mobile testing possible. Sony Electronics did not provide SIM cards and each broadcaster purchased enough SIM cards to complete their desired test plan. Each test plan had at least one control unit in the office for stationary reception. Then mobile units could venture out with news crews or other vehicles to test mobile reception of data.

Sony's Xperia phones are tested to work with Verizon, T-mobile and AT&T networks. Each broadcaster was allowed to choose their wireless carrier of choice (and obtain a SIM card) to complete these mobile tests. Data from ATSC 3.0 broadcasts did not interfere with the cellular connections so real time data logging could be achieved without artifacts of two different radios.

D. NRT Reporting Code

Sony's reports used JSON objects with agreed schema as defined above. The code for reports is shown below.

```
    HttpsURLConnection conn = (HttpsURLConnection)
url.openConnection();
    conn.setReadTimeout(15000);
    conn.setConnectTimeout(15000);
    conn.setRequestMethod("POST");
    conn.setDoInput(true);
    conn.setDoOutput(true);
    conn.setRequestProperty("Content-Type", "application/json" );
    conn.setRequestProperty("authorization", "Basic
YWRtaW46cGFzc3dvcmQ=");
    conn.setRequestProperty("User-Agent", "SonyAtsc3/Test2");

    JSONObject jo=new JSONObject();
    jo.put("date", getReceiveTimeInW3CFormat());
    jo.put("description", "Sony receiver test");
    jo.put("deviceID", "NVidia Shield");
    jo.put("fileName", filename);
    jo.put("operation", "create");
    jo.put("size", String.format("%d", size));
    jo.put("status","received");
    jo.put("signalStrength",String.format("%.1f",
signalStrength/1000.0));
    jo.put("signalQuality",String.format("%.1f",signalQuality/1000.0));
    jo.put("channelNumber",channelNumber);
    jo.put("speed",String.format("%d", speed));
    jo.put("nrtServiceID",serviceId);
    jo.put("nrtPlpID",String.format("%d",plp));
    jo.put("GPS coordinates",
String.format("%.5f,%.5f",latitude,longitude));
    jo.put("Pre-LDPC Error Count", String.format("%.3E",
ldpcber/10000000.0) );
    jo.put("Pre-BCH Error Count", String.format("%.3E",
ber/1000000000.0) );
    jo.put("Demod Lock", 1);
    for (int i=0;i<numPlps;i++) {
      String name=String.format("ALP %d Lock",i);
      jo.put(name, alpLockState[i]?1:0);
    }
    for (int i=0;i<numPlps;i++) {
      String name=String.format("ALP %d Baseband Packet Error",i);
      jo.put(name, pen[i]);
    }
    jo.put("Global Service ID", gsid);
    jo.put("# of PLPs", numPlps);
    jo.put("Battery Life", battery);
    String msg=jo.toString();
```

The return data is sent back to the Core Network. To enable reports, inside the Sony ATSC 3.0 Application, press LEFT ARROW for a few seconds to bring up the main menu. Then navigate:

1. MAIN MENU→FILES→TOGGLE LOOP→ENABLE
2. MAIN MENU→FILES→NRT REPORT→ENABLE

To separate out devices in and across markets, a deviceID pin with 4 digit number can be extended to the phone's assigned deviceID. The default is 0000. From NRT REPORT menu item swiping up or down increments and decrements this pin by 1. Example is Sony XQ-AQ62 0006. Then the report with show, for example, Sony XQ-AQ62 0003 if the pin is set to 0003. These deviceID's will persist through a power cycle, but will reset upon a Clear Cache of the Sony ATSC 3.0 Application. Kansas City uses deviceID extensions 1 -5, Detroit uses 6 – 10, Denver uses 11 – 15 and Indianapolis uses 16-20.

To receive large files (>35Mbyte), repair process needs as many partitions as the source transmitter provides. For very large files (GByte), set the interleaving on the Triveni GuideBuilder to use less source blocks at a time, then the receiver can handle files in GByte range. To increase the number of partitions on Sony phone, navigate to File menu and click FEC item. Swipe right to highlight TOGGLE NRT OFF then swipe up. The number of partitions will increase and show on the FEC log. Set to number to something very high and it will repair every source block separately.

The Sony APK software can detect when all files listed in the S-TSID have been received before initiating the loop timer to reset and re-capture the same file in the carousel. Loop time starts after all files received as listed in S-TSID (still need to set enable TOGGLE LOOP). Default and min time is 10secs but can be increased. The loop time is adjustable by highlighting the Toggle Loop and swiping up and down, it increments and decrements in units of 10. For files <35 Mbyte, a TOGGLE LOOP time of 300 seconds should be sufficient, but the loop time depends on the size of the file carousel (how long it takes to deliver all files). Testing used 300 seconds as that was sufficient for the files sizes chosen. Behavior is that a report is delivered if all files listed in the S-TSID are sent within the 300 seconds, otherwise the timer has to be increased.

As a result of reporting upon successful reception of files listed in the S-TSID, another report back is sent when all files in the S-TSID are received. For example, if 3 files of the same size are sent, each of those file delivery times will be reported in addition to the completion time of all 3 files. Upon success of all files received within 300 second timer and the return path report sent, all files will be erased and tried to download again for the next return path report.

VII. Results

There are 4 markets, Detroit, Kansas City, Denver and Indianapolis connected to HPE's Core Network. EW Scripps markets of Detroit and Kansas City use Triveni hardware which implemented the AL-FEC enhancements for extended packet loss coverage at 10% repair. Nexstar markets of Denver and Indianapolis use DigiCap hardware which did not implement AL-FEC enhancements by the time of this paper. This allows for a good comparison of the added protection using Repair Flow.

There are a few conditions of interest, which outline this chapter's sub-sections.

1. Time delay between sending a file from Core Network to delivery in the 4 markets
2. Signal Quality reports with stationary and mobile service
3. Added protection results using AL-FEC

Results are from file reporting to the Core Network, which is only done upon completion of the file being correctly received. Meaning, if there was packet loss and enough repair bits are available to fix, then a report is reported back to Core Network. If the file was delivered, but there was no repair flow or not enough repair bits are available to fix, then no report will be delivered for that file carousel. Subsequent file delivery will be used until a full correct file is received until a report is delivered back to the Core Network.

The Core Network has the ability to control the PHY configuration in each broadcaster station and for this trial, there are a few configurations to choose from which are listed in Table VII.1.

Table VII.1 PHY Operating Points

Multiplex option	Payload	Operating Point
Layered Division	2.69Mbps *Denver, *Indianapolis	-2.73dB SNR
	4.73Mbps *Detroit	0.46dB SNR
	6.77Mbps	2.93dB SNR
Time Division	0.89Mbps	-2.31dB SNR
	1.33Mbps	0.33dB SNR
	3.13Mbps *Kansas City	5.79dB SNR

Detroit chose Layered Division Multiplexing with 4.73Mbps payload sizes, Kansas City chose Time Division Multiplexing at 3.13Mbps, Indianapolis and Denver choose Layered Division Multiplexing at 2.69Mbps. Differences in time delivery depend on file size which was different in each market, and changed depending on broadcaster interest.

- Denver datacasting is placed on the core PLP and assigned 2.69Mbit/sec payload datarate, but only using 2.00Mbit/sec. Denver is sending small files in this payload, so delivery times are fast.
- Detroit datacasting is placed on the core PLP and assigned 4.73Mbit/sec payload datarate, but only using 2.00Mbit/sec. Detroit is sending a variety of files with different sizes and delivery times vary as expected.
- Kansas City datacasting is placed on the core PLP and assigned 3.13Mbit/sec payload datarate, but only using 2.00Mbit/sec. Kansas City is sending a variety of files and delivery times vary accordingly.
- Indianapolis datacasting is placed on the core PLP and assigned 2.69Mbit/sec payload datarate, but only using 2.00Mbit/sec. Indianapolis is also sending small files so delivery times are fast.

A. Delivery Times

There are a variety of files being sent in Detroit, and in 24 hours the delivery times vary from 2 seconds to around 21 seconds for the files with a tight distribution of delivery times for each file as shown in Figure VII.1. There are 6 files being delivered,

1. 5.26Mbyte file taking around 21 seconds
2. 2.66Mbyte file taking just over 10 seconds
3. 0.63Mbyte file taking around 2.5 seconds
4. 1.44Mbyte file (1) taking just below 6 seconds
5. 1.44Mbyte file (2) taking just below 6 seconds
6. 1.44Mbyte file (3) taking just below 6 seconds

The combined time is around 51 seconds to receive all files. After reception of all files, the receiver timer of 300 seconds (5 minutes) erases all files and the Sony phone ATSC 3.0 application tries to re-capture them again to generate the next report.

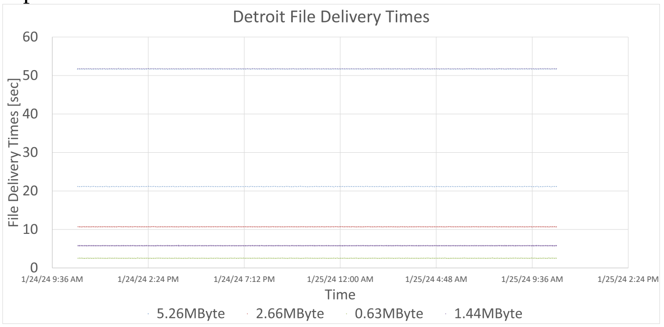


Figure VII.1 Detroit Delivery Times

Kansas City also has a variety of files being delivered, but these have Repair Flow applied. Delivery times correlate to file size where smaller files are received faster as shown in Figure VII.2. Kansas City have Repair Flow enabled, so packet loss is repaired before reports go back to the Core Network. Delivery time variance is very tight, graphs look like straight lines.

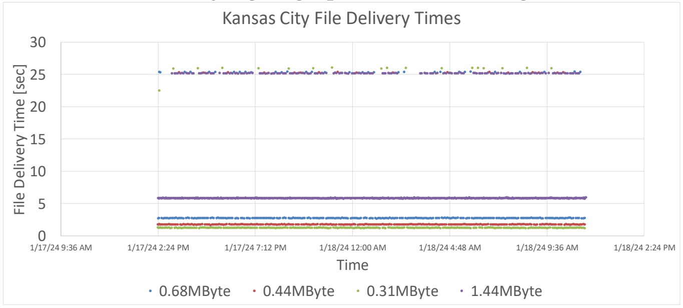


Figure VII.2 Kansas City Delivery Times

Denver also has 3 files being delivered, 60kByte, 120kByte and 170kByte lengths. Denver does not have Repair Flow operation enabled and the distribution of delivery times shows a large difference in sigma distribution of times as shown in Figure VII.3. Most of these small files are delivered in less than 5 msec, but without Repair Flow AL-FEC, correct file reception relies upon the file carousel to deliver the file again for correct reception. This greatly affects time of completed file reception and confidence of reception. Outliers go as far as 25msecs, 10x longer than average (2msec) for correct file reception.

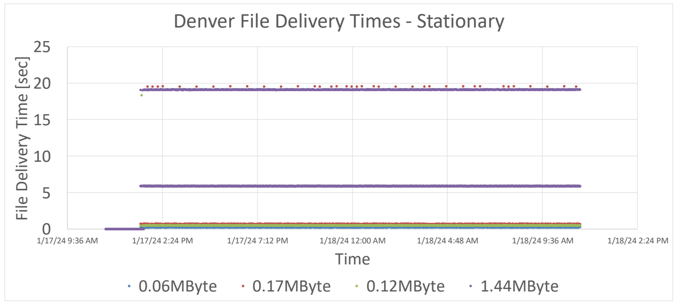


Figure VII.3 Denver Delivery Times

B. *Signal Quality Reports*

Stationary reception will show variations of Received Signal Strength Indicator (RSSI) and Signal to Noise Ratio (SNR) over time. For 24 hours, the results in Detroit are shown in Figure VII.4.

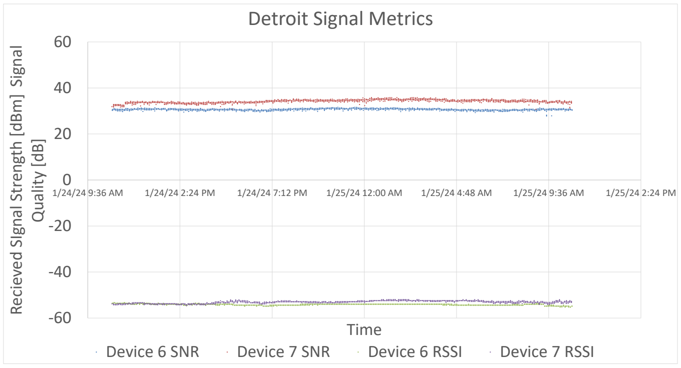


Figure VII.4 Detroit Signal Quality

Not much variation, but stronger received signal strength can be seen at night, a little interesting. There is very consistent reception values of signal level and quality (SNR) for every report delivered. Variance is low.

For 24 hours, the results in Kansas City are shown in Figure VII.5.

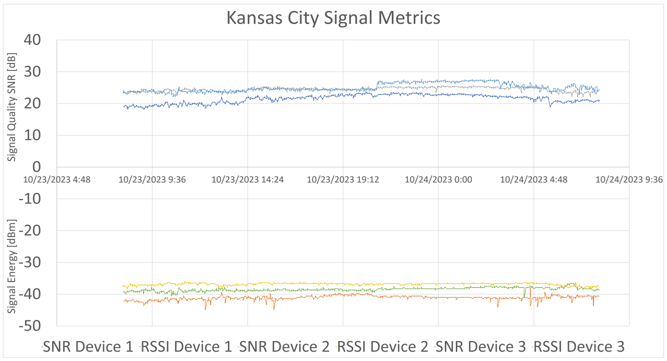


Figure VII.5 Kansas City Signal Quality

For 24 hours, the results in Denver are shown in Figure VII.6.

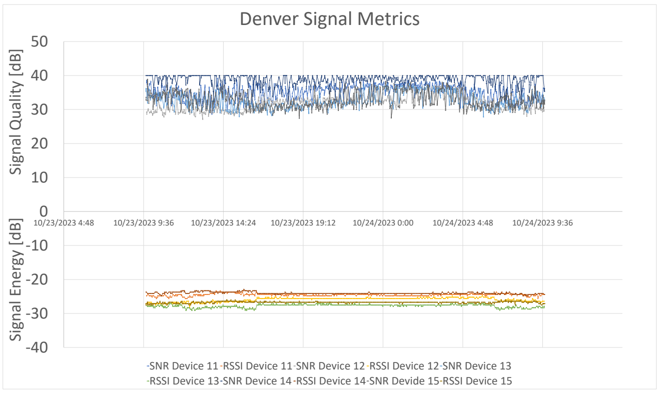


Figure VII.6 Denver Signal Quality

For 24 hours, the results in Indianapolis are shown in Figure VII.7.

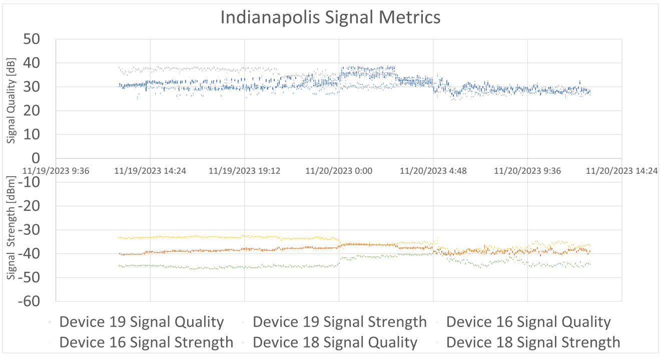
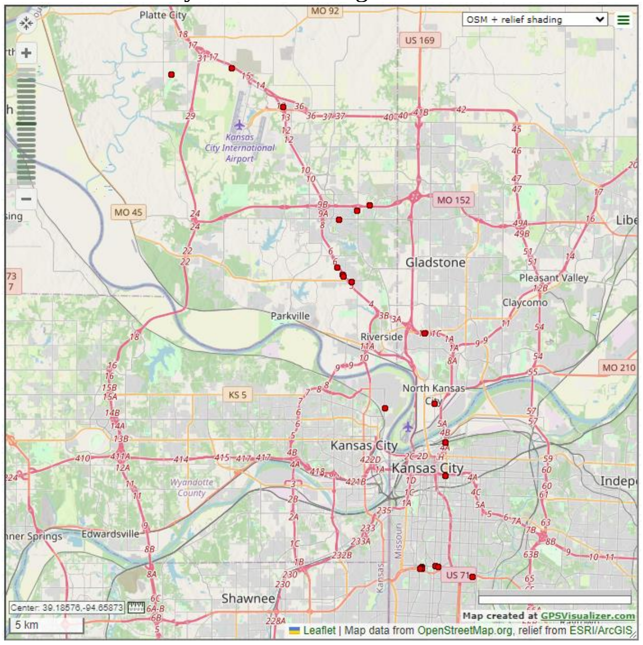


Figure VII.7 Indianapolis Signal Quality

Stationary reception is very stable and strong (when antenna amplifier is used) over 24 hours in multiple markets. There is some variation between devices, but they all are relatively consistent in their reporting.

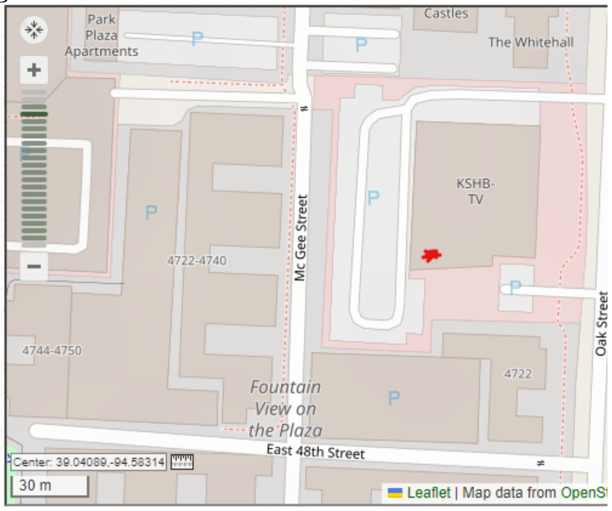
Reports are only delivered after full file reception, plotting mobile signal quality will be a bit sparse. An example of a drive home in Kansas City is shown in Figure VII.8.



Source: https://www.gpsvisualizer.com/

Figure VII.8 Kansas City Mobile Reports

Stationary reports are located at the station office as shown in Figure VII.9.



Source: https://www.gpsvisualizer.com/

Figure VII.9 KSHB Stationary location

Since Kansas City employs Repair Flow, there is a question of difference between them? Results show no discernable difference in delivery times and reinforces the point of Repair Flow offering an 'insurance of delivery' to correct a percentage of possible errors.

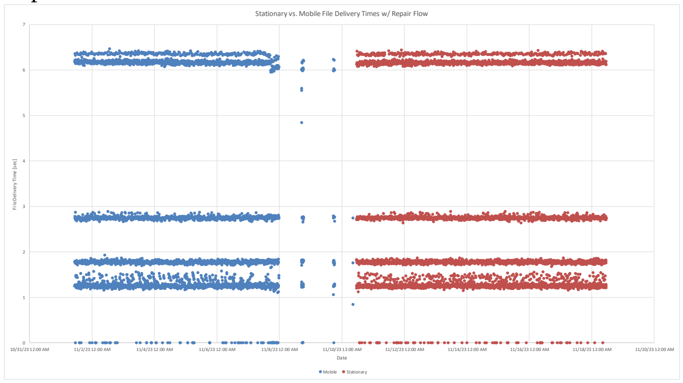
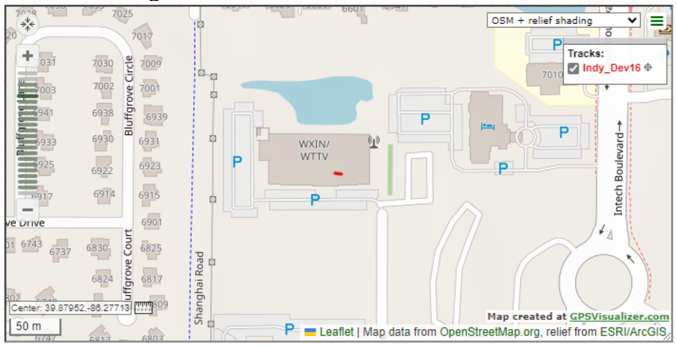


Figure VII.10 Mobile vs. Stationary

Figure VII.10 is plotted over days, not hours and the mobile device testing was first. Afterwards, the stationary device was turned on and plotted in red. Note, the mobile reports (blue) drop off at the end of their testing days due to zero percent battery life. Sony receivers used the phone battery plus a possible portable power pack which was only designed to last 6 hours. Files were missed only when battery life expired, otherwise it was hard to find when any files were missed.

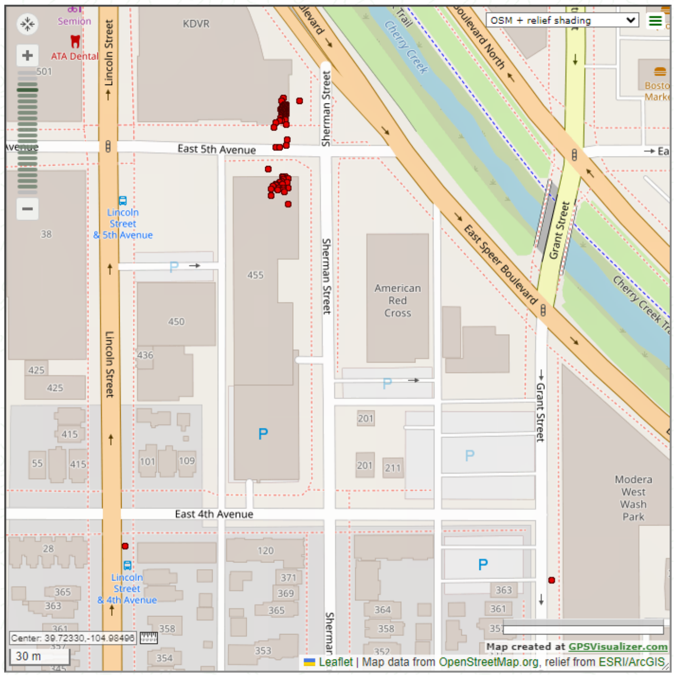
Indianapolis station office reports shows consistent location a shown in Figure VII.11.


OSM + relief shading
Tracks:
Indy_Dev16
WXIN/ WTTV
Bluffgrove Circle
Bluffgrove Court
Shanghai Road
Intech Boulevard
Center: 39.87952,-86.27713
50 m
Map created at GPSVisualizer.com
Leaflet | Map data from OpenStreetMap.org, relief from ESRI/ArcGIS

Source: https://www.gpsvisualizer.com/

Figure VII.11 Indianapolis stationary location

Denver stationary reports show a little variance in location (Figure VII.12), likely due to GPS reflections into the phone.



Source: https://www.gpsvisualizer.com/

Figure VII.12 Denver Stationary location

C. AL-FEC Protection

Raptor-Q FEC with 10% repair bits offered enough protection in Kansas City to see that only 2 file carousels were needed to completely recover all files. This adds confidence and robustness to customers looking to send files only a few times. Delivering files without repair bits still shows success in these 4 markets, but difficult channel conditions might not have been fully exercised.

Repair flow operation could be an option for business plans. Customers may choose to not use repair flow and just send the file many times until enough confidence is built that the file is received. This may depend on file size, as smaller files are not likely to suffer as much packet loss as large files. Security might also be a factor in repair flow use, where firmware or highly sensitive data that need guarantee of delivery within a certain timeframe justifies use of the repair flow.

Either way, the above results show repair flow adds protection to packets and can benefit from packets being lost up to 6 seconds with 10% repair. This of course can be changed and increased for even more protection of files. Even though Sony Xperia phones were used to perform the Raptor-Q algorithm, memory constrained platforms like Raspberry Pi can also run the Raptor-Q algorithm in small pieces which are signaled in the extended ATSC A/331[1] signaling described in Section 5.

VIII. Conclusion

Adding AL-FEC repair bits extends packet loss coverage beyond what is available in Source flow. Reliable reception was demonstrated in 4 markets for datacasting to all devices without impacting existing TV services. Small files allowed more accurate mobile testing of signal conditions while larger file delivery showed delivery time latency according to file size.

Business use-cases can offer options to customers whether they want robust delivery, guaranteed times of delivery or if they have small files that can just run in a file carousel to update fielded devices. Datacasting with ATSC 3.0 offers new use-cases for private and public entities where broadcasters can expand their core support function of society…delivering information.

Acknowledgment

The authors would like to thank our colleagues at Sony Electronics, Inc. for providing Xperia Phones, Hewlett Packard Enterprise, Heartland Video Systems, and our broadcast friends at EW Scripps broadcasters at WMYD and KMCI, Nexstar broadcasters at WTTV and KDVR. Their help in this field test was instrumental to ensure proper testing and success.

References

[1] ATSC: "ATSC Signaling, Delivery, Synchronization, and Error Protection," Doc. ATSC A/331:2023, Advanced Television Systems Committee, Washington, D.C., 13 December 2023

Luke Fay is a Senior Manager Technical Standards for Sony Electronics Home Entertainment & Sound Products - America. Currently he is involved with the development of the next generation of broadcast television in a variety of standards organizations and their efforts to educate members of the new possibilities available with ATSC 3.0. He has over 20 years of experience in digital communications systems engineering and receiver design.
He received a BS in Electrical Engineering from University of Arizona and an MS degree in Electrical Engineering from National Technological University. He has been granted over 20 patents in the area of Digital Signal Processing.
He is currently serving as Chairman of the Advanced Television Systems Committee (ATSC) Technology Group 3 (TG3). He is also the recipient of the 2015 Bernard J Lechner Award for technical and leadership contributions to the ATSC. He became a SMPTE Fellow in 2018.

Graham Clift is a Principal Hardware Engineer for Sony Electronics Home Entertainment & Sound Products - America.