ELUVIO CONTENT FABRIC

Decentralized, Programmable, Just in Time

IBC 2018

http://eluv.io

WHAT IS IT AND WHAT ARE THE BENEFITS?

- A software platform for highest quality, format agnostic, low latency content distribution at min cost (10X less)
- Eliminates redundant network and storage use and self scales through a novel decentralized design
  - Content protected from the infrastructure end-to-end
  - Bandwidth, compute, and storage resources can extend infinitely
  - Efficient and flexible content personalization with feedback loop
  - Internet scale marketplace of content, user engagement and sponsorship
  - Users transparently share their data (or not)
  - Exposure and elimination of content "counterfeits“
  - Simplification of complicated content supply chains and localized siloed workflows

PRESENTATION OUTLINE

- History of the Internet & A New Content-Centric Approach
- Content Fabric Design Principles & Technology
- Demonstration Scenarios for the Distribution to Global Viewers
    - Dynamically Rendered Multiple Language Versions
    - High bandwidth, low latency delivery
    - Instant Repair without Repackaging or Redistribution
    - Personalized and Dynamic Media via Bitcode
    - Fast content proofs to identify corrupted / invalid / counterfeit content
    - Scalable Smart Contracting such as Advertising Marketplaces, Instant Licensing, and Electronic Sell Through
- Conclusions

BACKGROUND
CONTENT ON THE INTERNET

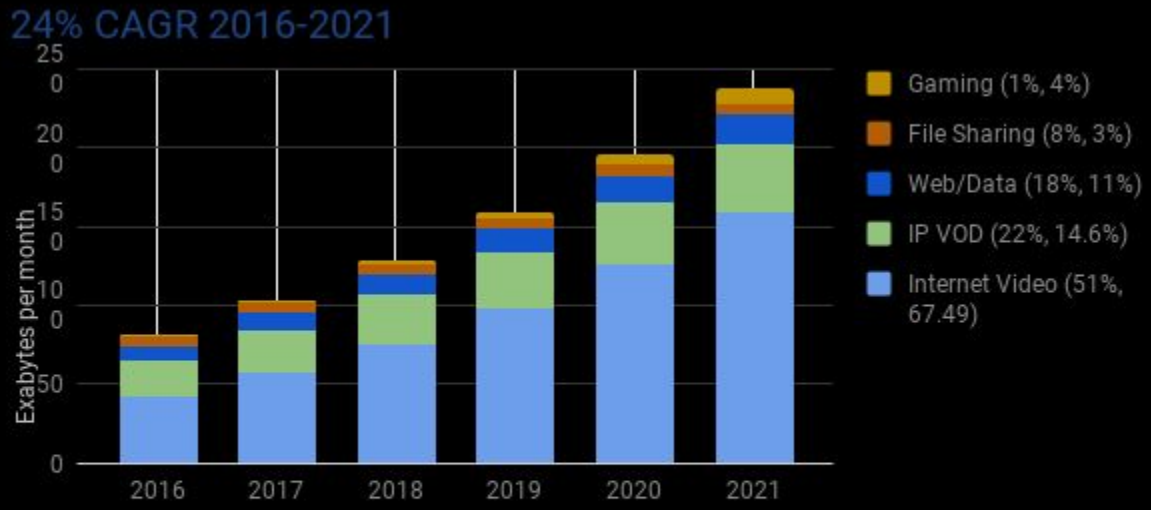
Today's Internet is dominated by Content (not data)

"In the 21st century it is the flow of attention, not information (which we already have too much of) that matters."

"Attention (is the) crucial resource of the digital economy..."

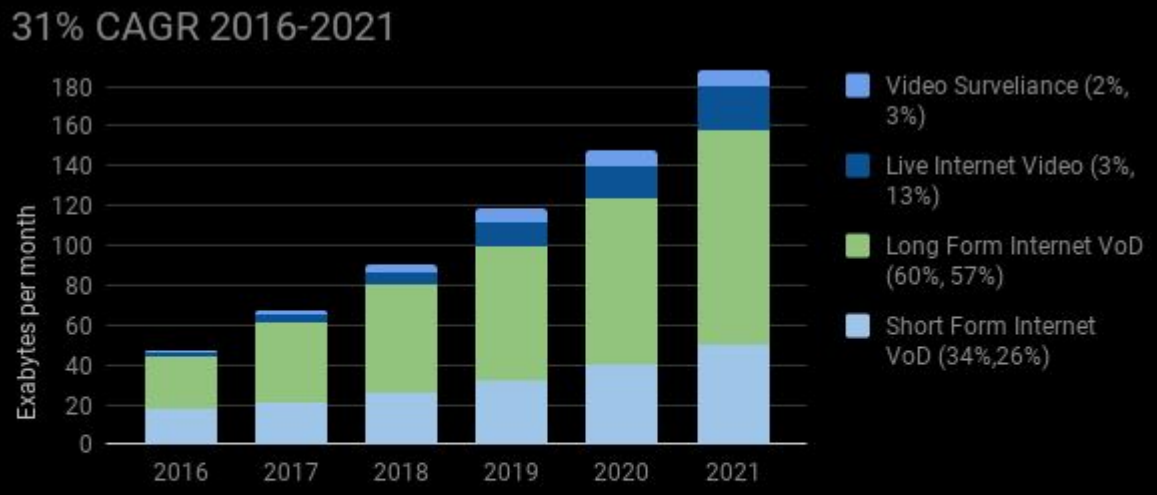
--Zenep Tufecki, Univ of North Carolina

82% of all Internet Traffic will be Video by 2021!



Figures (n) refer to 2016, 2021 traffic shares. Source: Cisco VNI Global

“Live” Internet Video is growing the fastest



Figures (n) refers to 2016, 2021 traffic shares. Source Cisco VNI

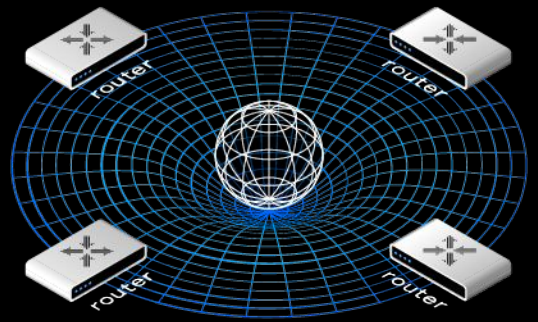
Yet the Internet architecture has not changed ...

Still based on the original host-to-host communication design of 1970s:

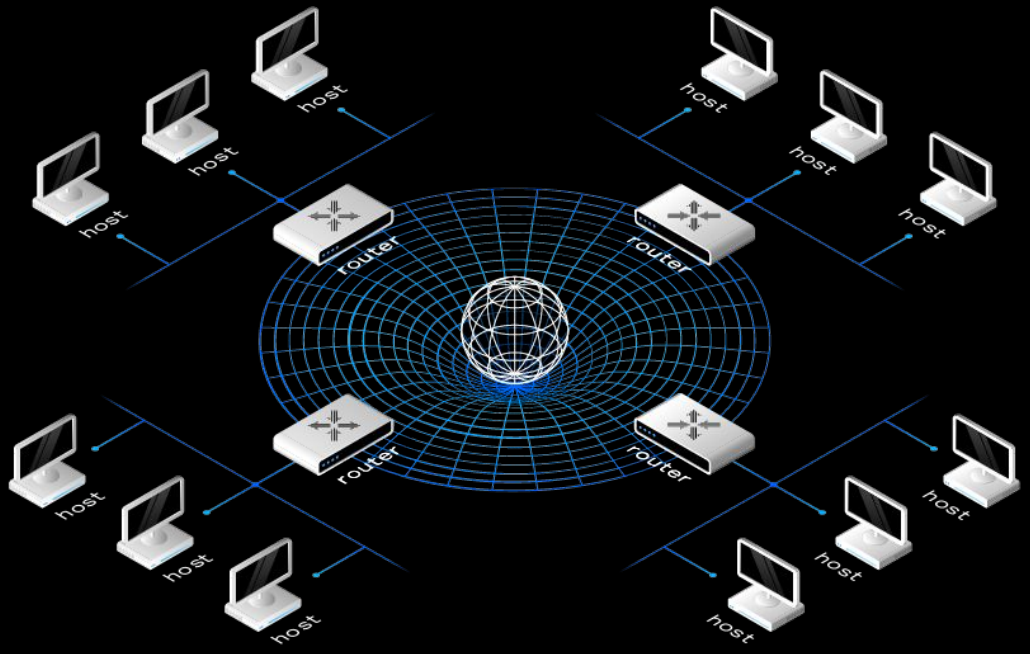
- Lacks ability to serve content natively
  - Large and expensive distribution services (CDNs)
  - Complex workflows with **much duplication of content**
  - Uneven quality and latency
- Lacks a common content security framework
- Lacks native commerce and transparency

Vendors benefit from these deficiencies while content owners struggle competing.

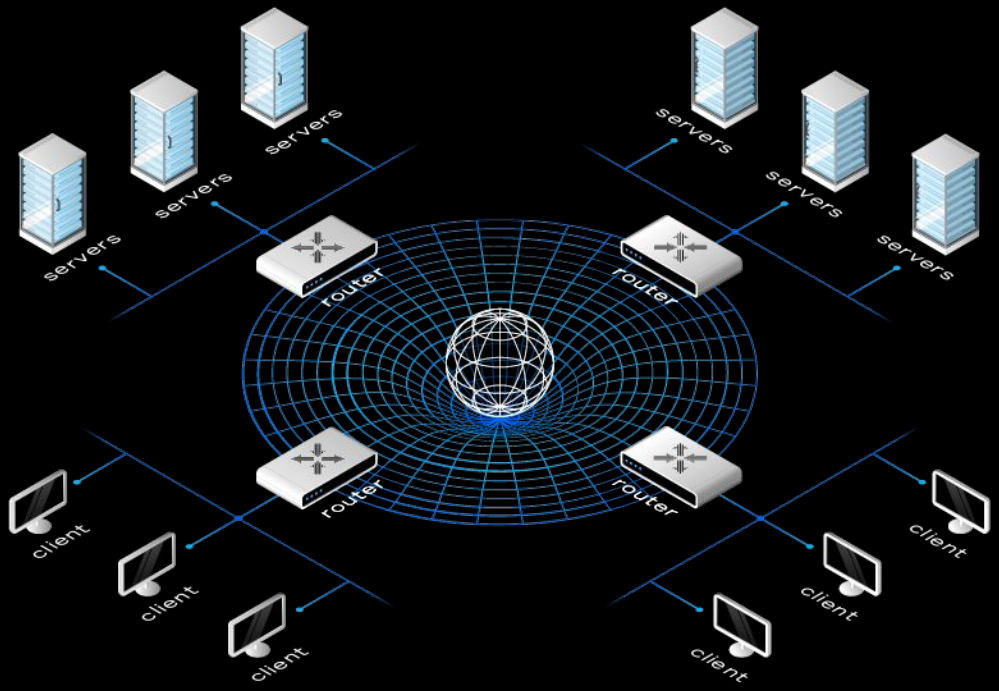
THE INTERNET BEGAN AS IP ROUTING



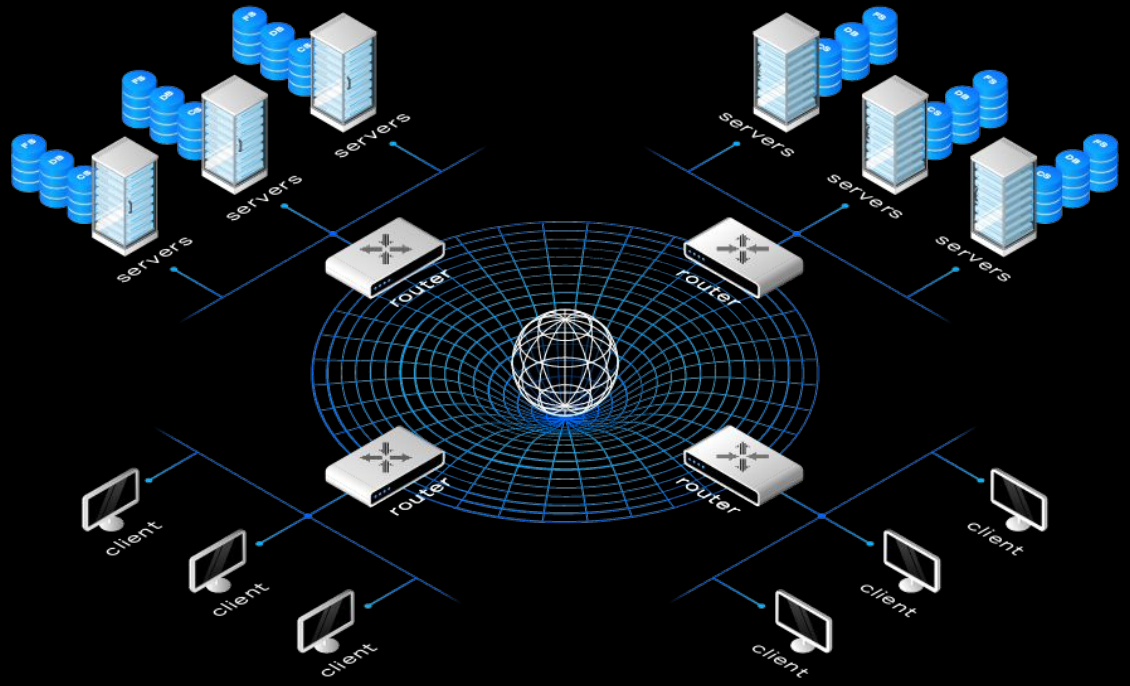
FOR COMMUNICATION BETWEEN HOSTS



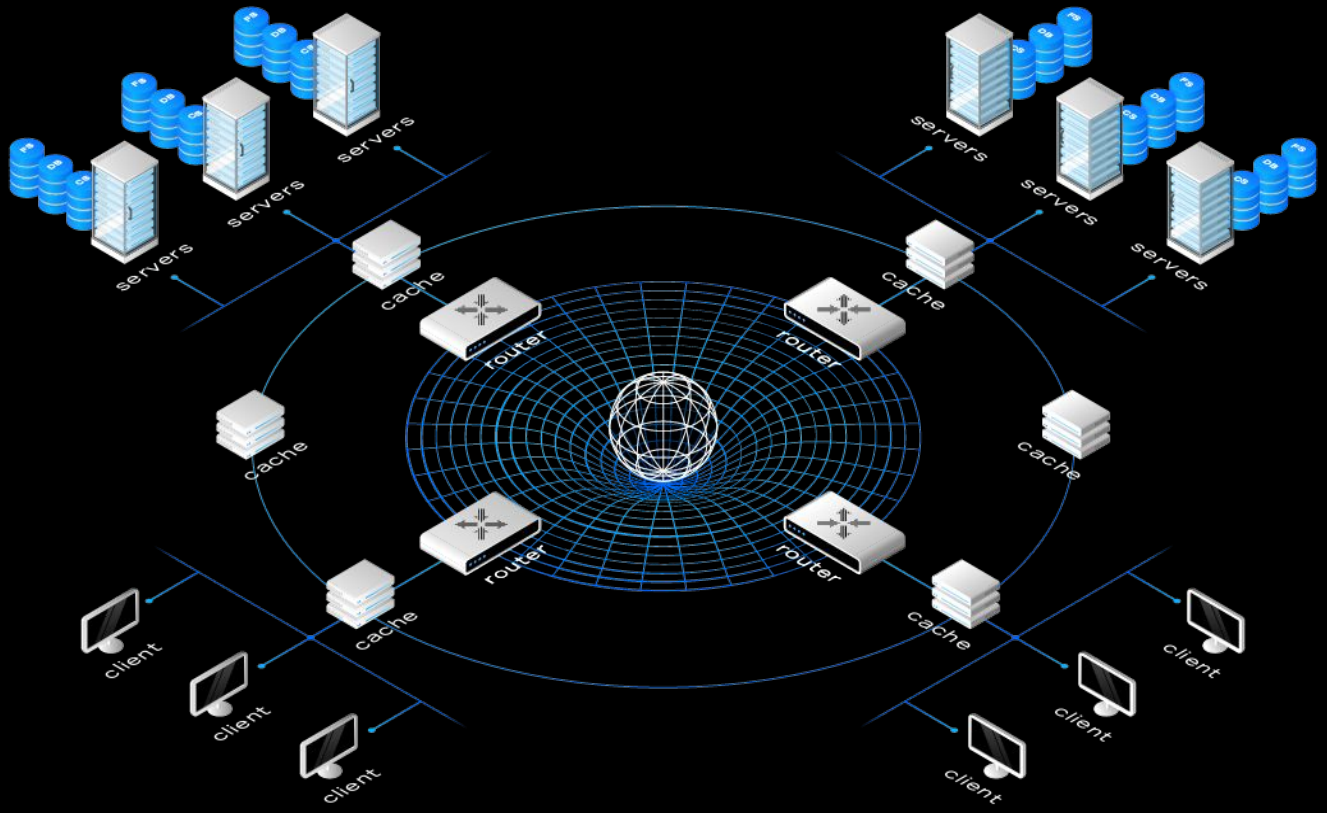
THEN BECAME A CLIENT-SERVER ARCHITECTURE



AND SERVERS GOT ALL THE DATA (CONTENT)



THEN WE HAD TO DEPLOY EDGE CACHES



CONTENT and the INTERNET

- The Internet was designed to enable two ‘hosts’ to have a conversation
- Things have changed
  - today 80+% of all Internet traffic is access to ‘digital content’ (mostly ‘video content’)
- Van Jacobson, 2006: "Content Centric Networking"
  - Academic initiative to replace IP with content protocols
  - Remove redundant core bandwidth usage
- The technology ecosystem in 2017 far outstripped his vision:
  - A.I. / M.L., blockchain ledgers, GPUs/CPUs/TPUs, cryptographic theoretical frameworks
  - Opportunity is much bigger to improve the full supply chain for content provision and distribution

OPPORTUNITY

There are many design principles (some old, some new) that can help make the Internet ‘digital content-centric’.

… applying “content networking” principles

… as a software overlay

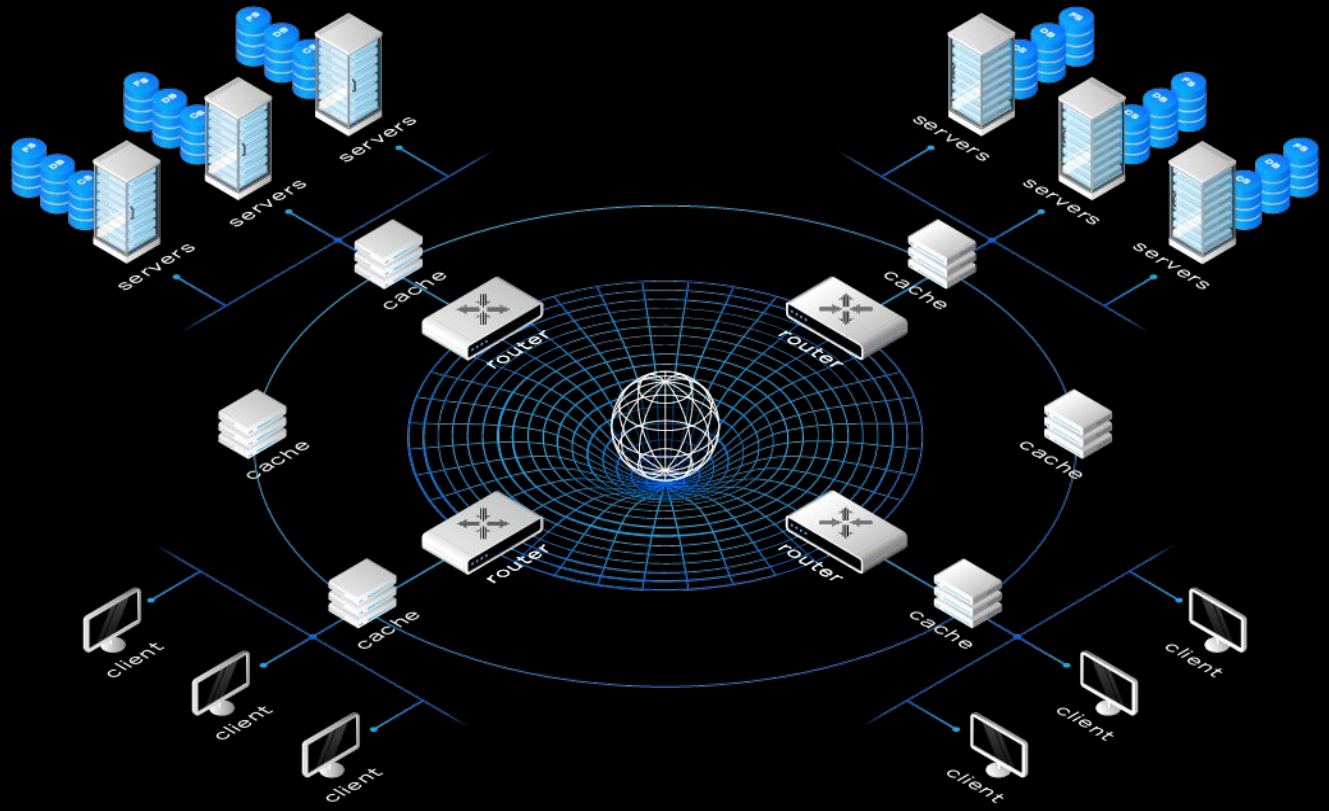
… via a new decentralized, distributed and programmable content platform

… in a single stack

… addressing the ***scalability***, ***protection*** and ***commerce*** problems natively.

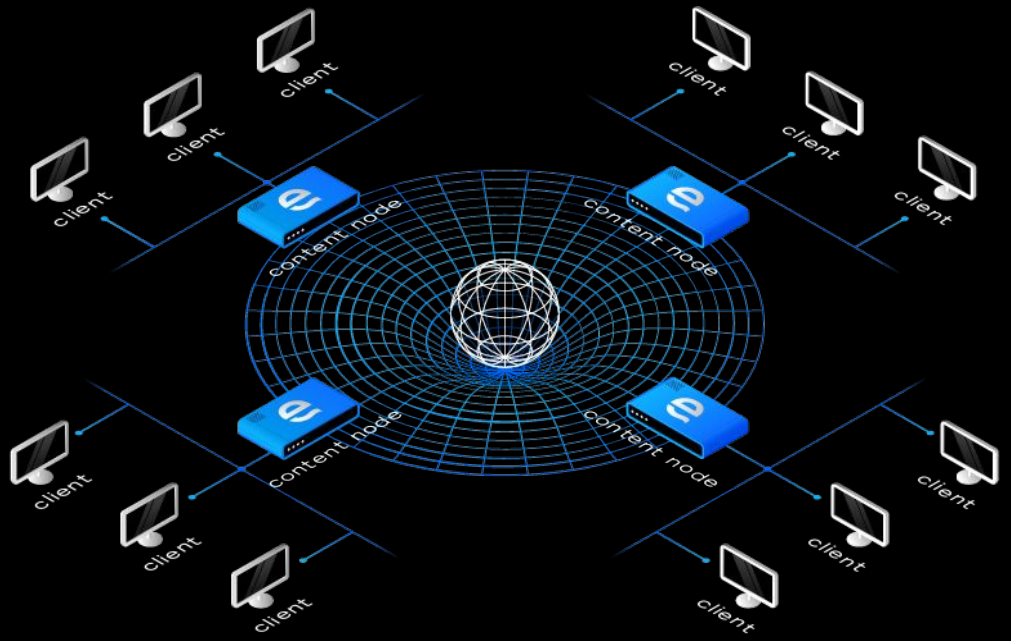
But fundamental innovation is needed.

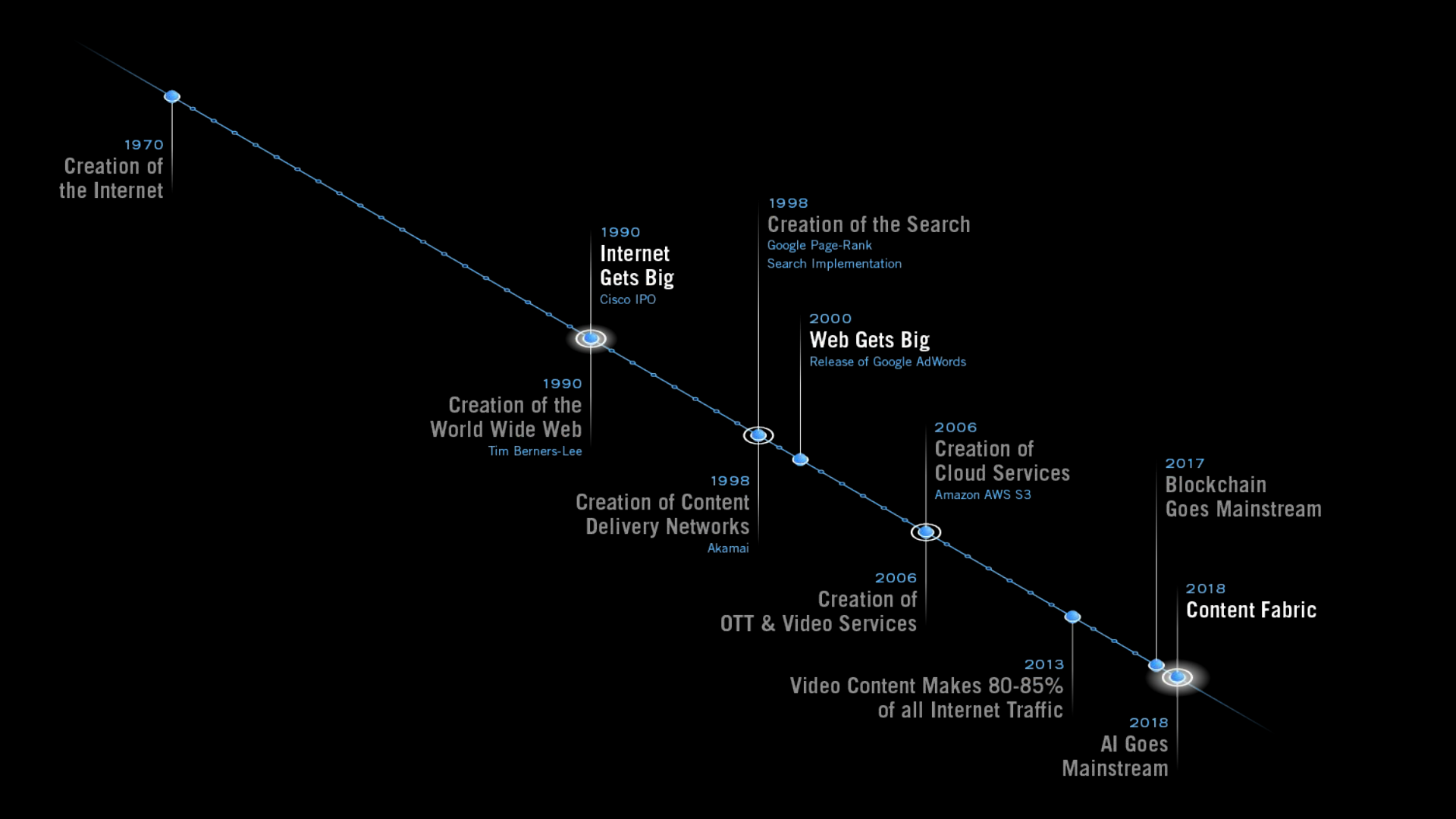
CONTENT-CENTRIC CHANGES THIS ...



INTO THIS ...

MAXIMALLY EFFICIENT, INFINITELY SCALABLE, PROGRAMMABLE, SECURE




1970
Creation of the Internet
1990
Internet Gets Big
Cisco IPO
1990
Creation of the World Wide Web
Tim Berners-Lee
1998
Creation of the Search
Google Page-Rank Search Implementation
2000
Web Gets Big
Release of Google AdWords
1998
Creation of Content Delivery Networks
Akamai
2006
Creation of Cloud Services
Amazon AWS S3
2006
Creation of OTT & Video Services
2013
Video Content Makes 80-85% of all Internet Traffic
2017
Blockchain Goes Mainstream
2018
Content Fabric
2018
AI Goes Mainstream

HOW IS THIS ACCOMPLISHED?
PRINCIPLES AND TECHNOLOGY

PRINCIPLES

- Content Internet Overlay
  - Single Software Stack on hosts (“Content Nodes”) for content and metadata
  - Distributed and Decentralized - “nothing is shared”
- Content and Metadata are stored natively and “once”
  - All end-products are dynamically rendered from the original source, on demand
- Content nodes are not trusted & content is protected (encrypted)
  - But are measured and rated publicly for behavior and performance
- Access is mediated through blockchain transactions
- Programmable
  - Programmable plug-in layer for any I/O format
  - Blockchain transactions obey programmable contracts → implements business use cases

ELUVIO TECHNOLOGY

Decentralized Content and Metadata & Just-in-Time, Personalized Composition

- Fast distributed hash structure for content and metadata
- Just-in-time rendering of content to all end versions
- No duplication of bytes on network or storage)
- Dynamically loaded “programmable build instructions” for maximum flexibility & personalization

Fast, Ultra efficient Routing and Distribution of Content

- Continuous Machine Learning maximizes delivery bandwidth, minimizes time to first byte, minimizes core bandwidth, and adapts to load and failures
- Fast distributed hash structure locates content globally with <1 RTT lookup time
- ML predictive caching combines with rendering from source for superior caching efficiency

Decentralized Security and Trust Framework

- Embedded blockchain ledger for access control +programmable ‘smart’ contracts
- Content version history backed by blockchain transactions and verification proofs
- Content encryption is trustless (allowing 3rd party infrastructure participation!)

CONTENT, METADATA AND BITCODE

Content, Metadata and Bitcode

All stored as “parts” (chunks of bytes) on hosts running content fabric (“nodes”)

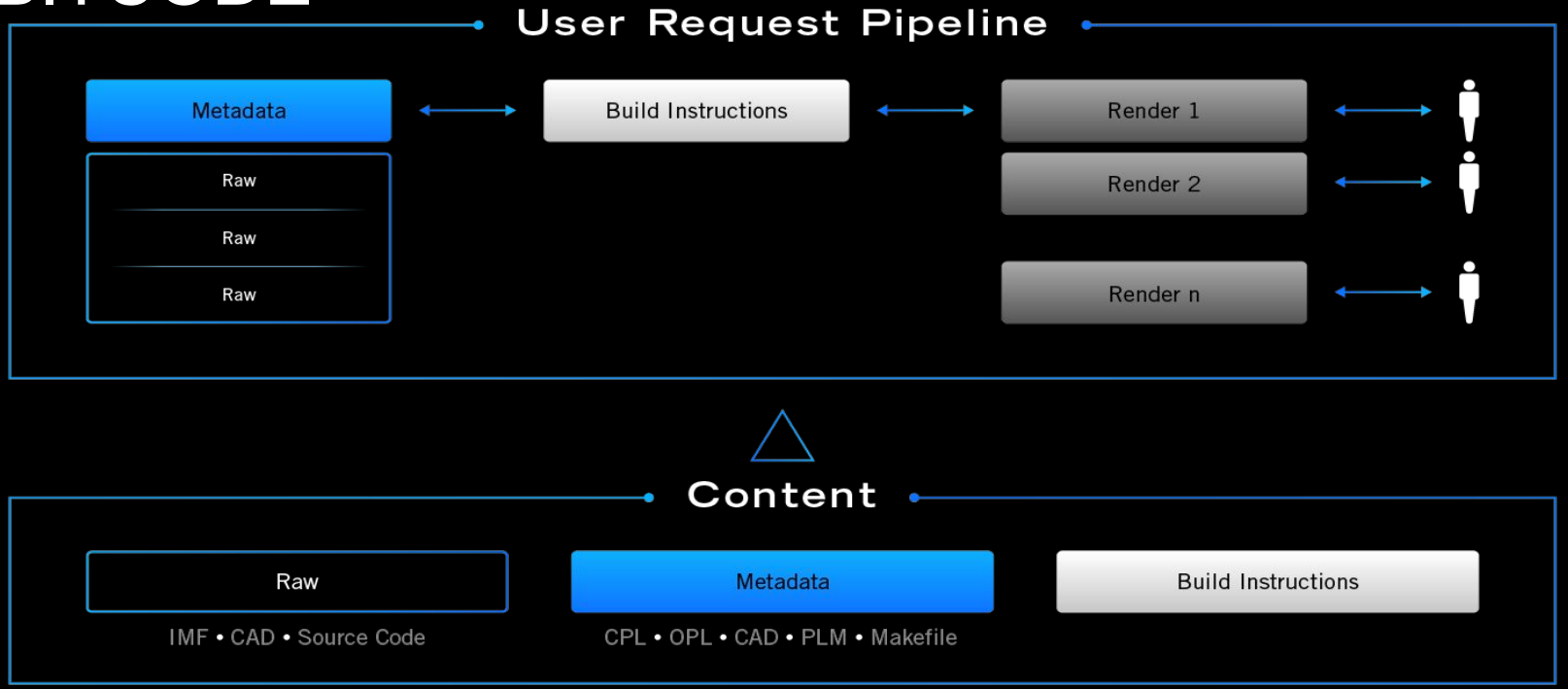
Referred to by a content object structure consisting of hashes of these parts (no duplicate storage of parts common to multiple objects)

Composed together on-the-fly at time of access/consumption

Each object is backed by a blockchain contract recording its version change history, “proof” (see next) and hashes

Authenticity verified on access using a fast proof that verifies the hashes of the constituent parts

IN FABRIC : CONTENT, METADATA AND BITCODE



ROUTING AND DISTRIBUTION

Novel Content Routing and Distribution

Parts are distributed and located on fabric nodes via a novel ultra-fast, decentralized hashing algorithm (< 1RTT)

Client requests for content route to a first node (egress), and if needed, to a second node (origin) having the part

Chosen routes maximize delivery bandwidth and minimize time to first byte using a novel continuous Machine Learning algorithm that “learns” the fastest, lowest latency path

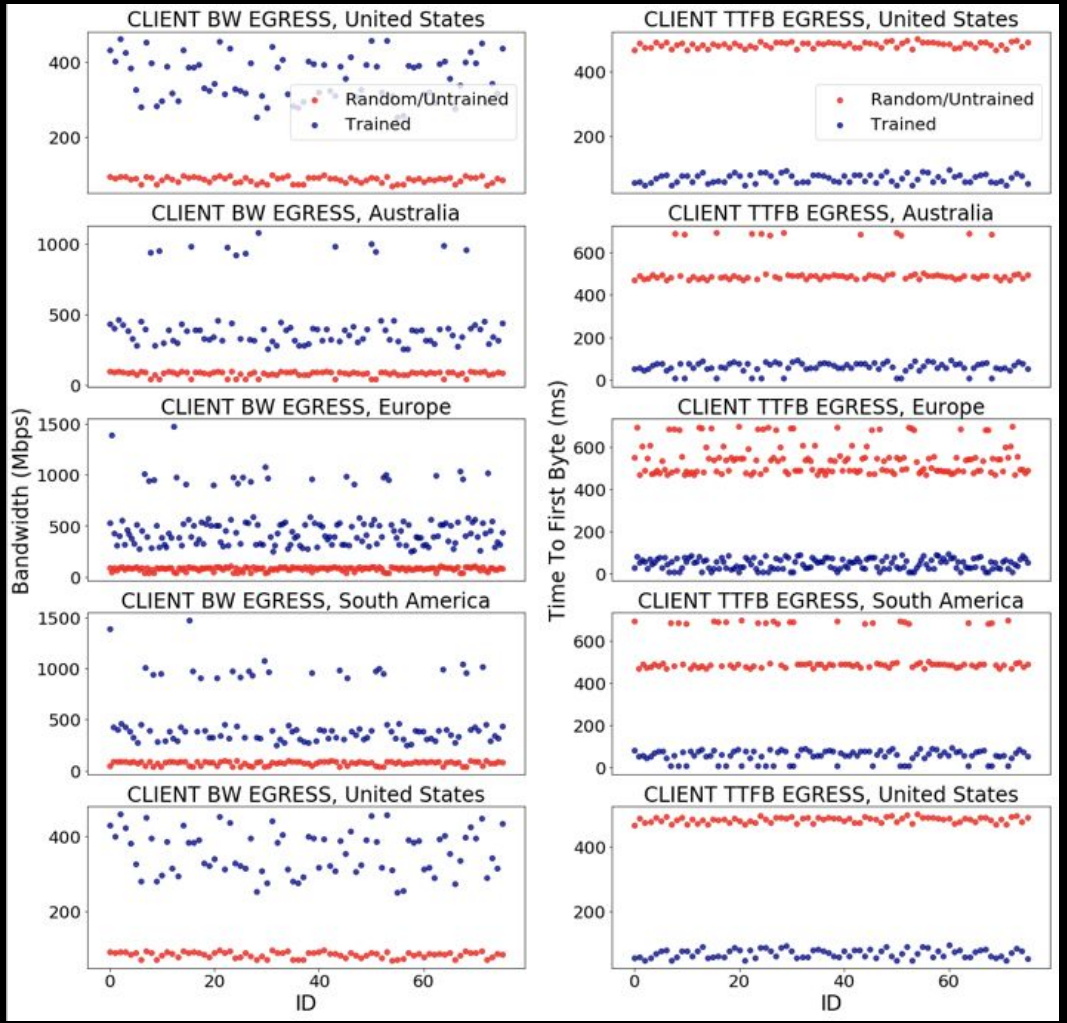
Popular parts are cached when served according to a novel ML predictive caching algorithm that exceeds today’s predictive caching in hit rate and bytes saved (5x)

ML Content Routing Results

Client Delivery Bandwidth and Time to First Byte for all Egress Nodes

Blue - Trained
Red - Random

Content Fabric on 52 nodes
5 continents:
AU, AS, EU, NA, SA
208 clients

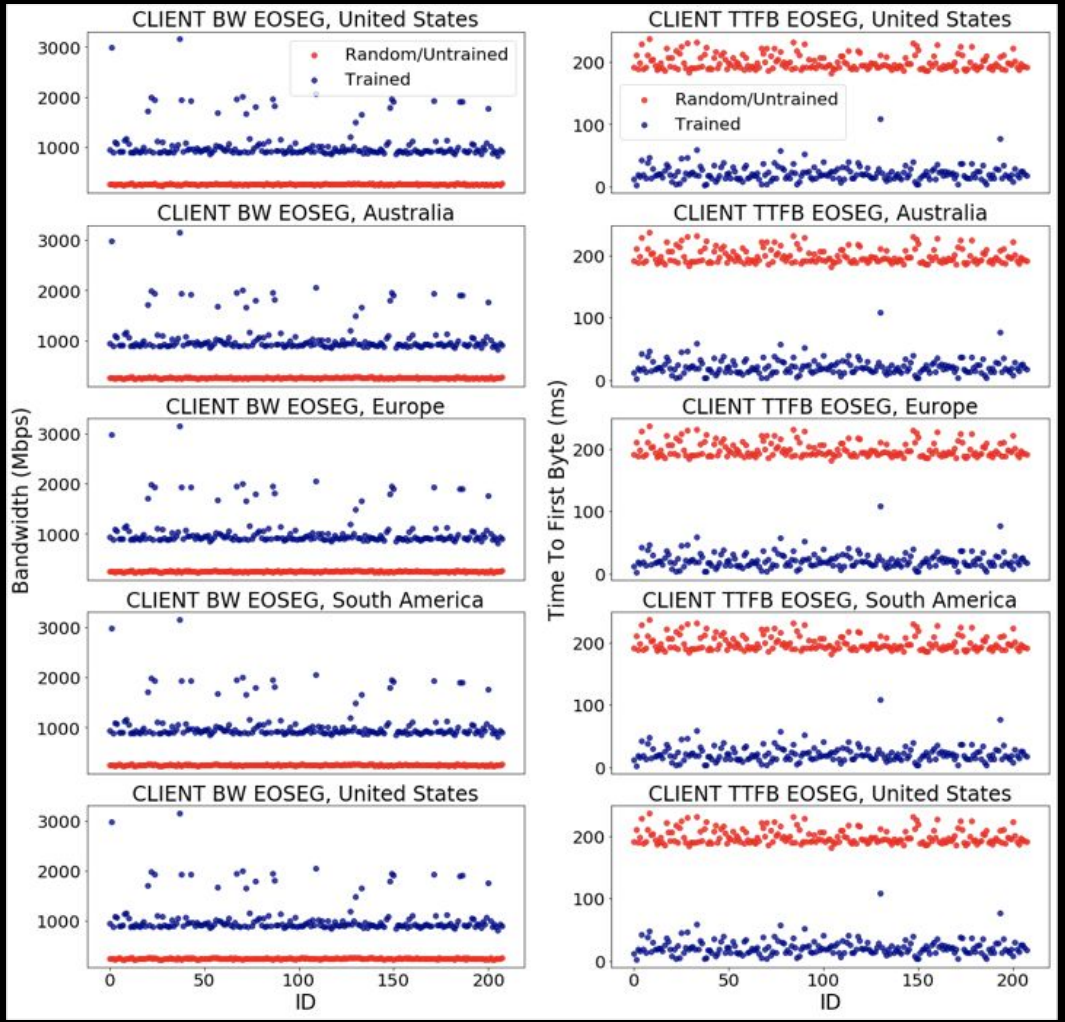


ML Content Routing Results

Client Delivery Bandwidth and Time to First Byte for all Egress-Origin Segments

Blue - Trained
Red - Random

Content Fabric on 52 nodes
5 continents:
AU, AS, EU, NA, SA
208 clients

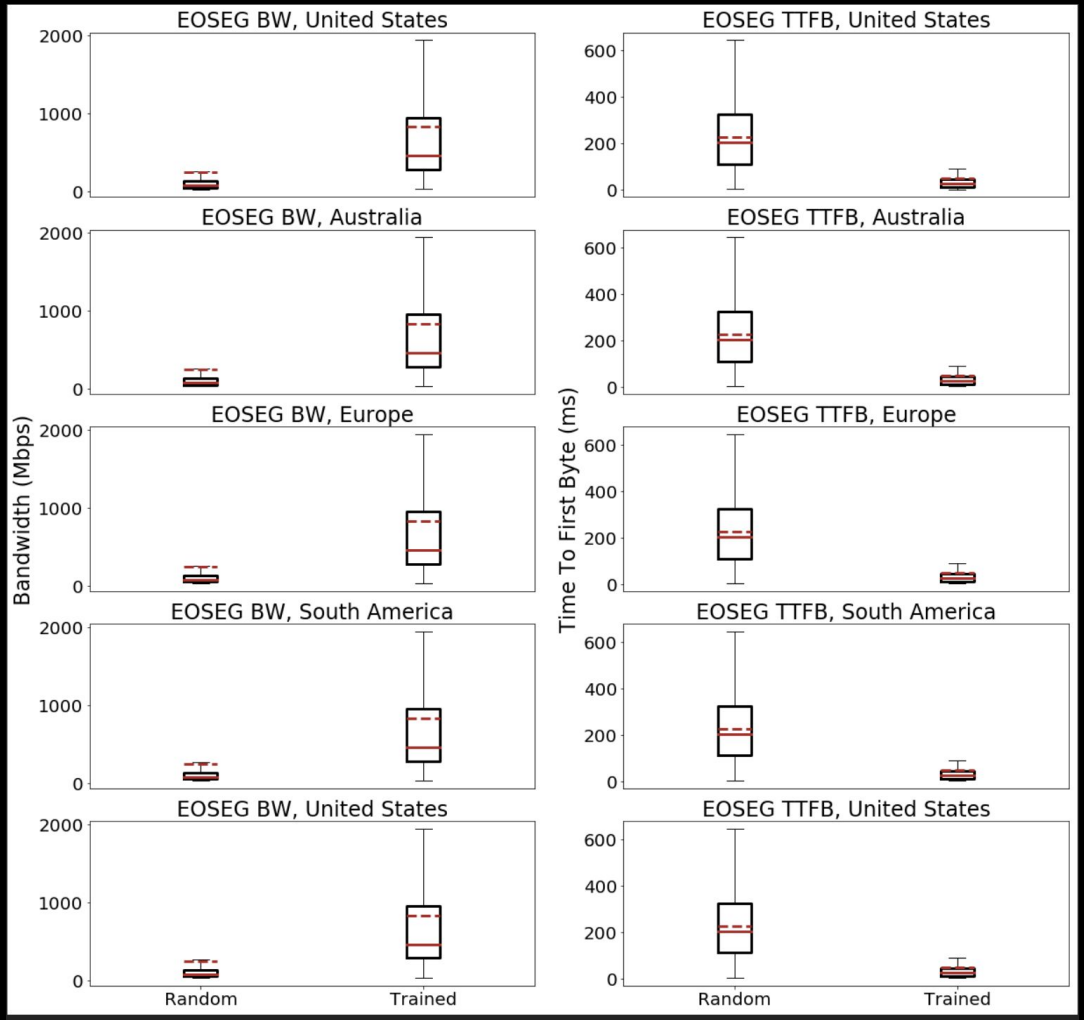


ML Content Routing Results

Client Delivery Bandwidth and Time to First Byte for all Egress-Origin Segments

Left - Random
Right - Trained

Content Fabric on 52 nodes
5 continents:
AU, AS, EU, NA, SA
208 clients



ML Predictive Caching Results

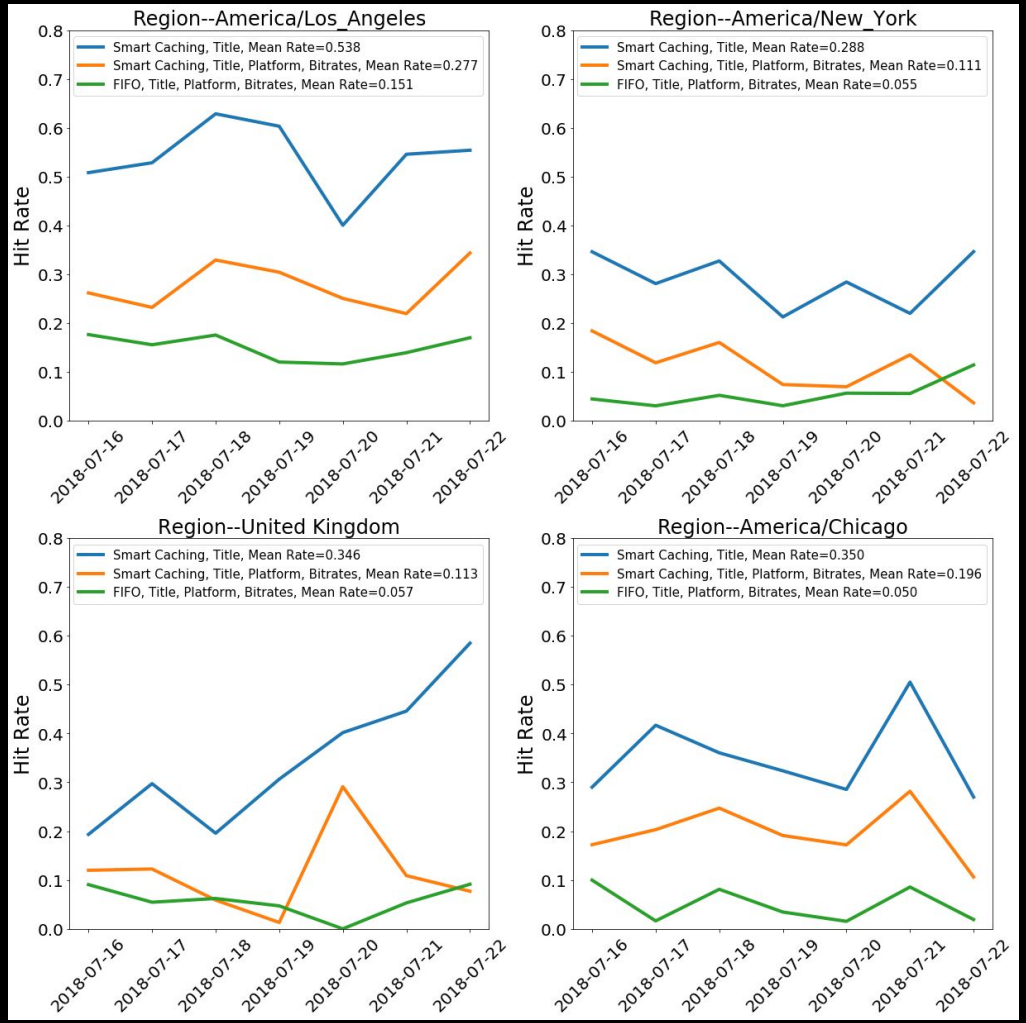
Trained/Tested Using Screeners Audience Data for One Year from Studio

12717 titles, 121 countries, 4159 cities,
Max streaming hours in one day: 5965
Min streaming hours in one day: 163

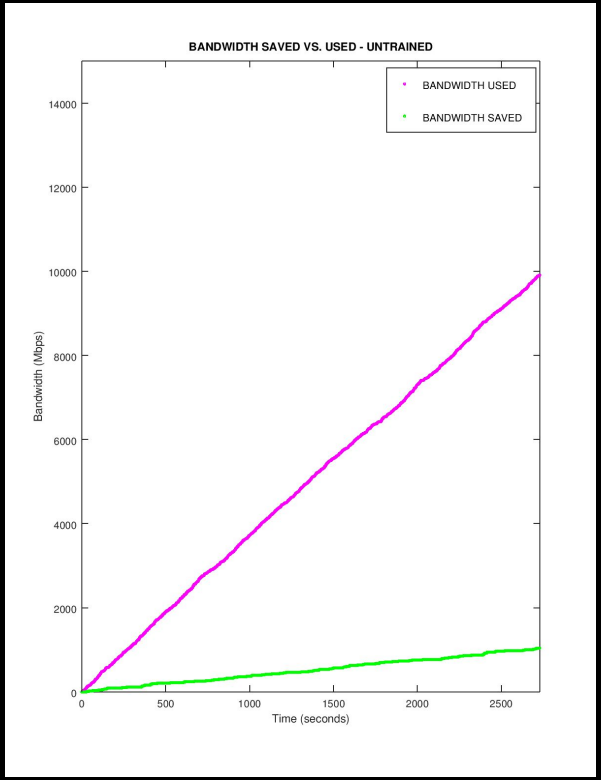
Blue - Eluvio Smart Predictive Caching (Needs Title only)

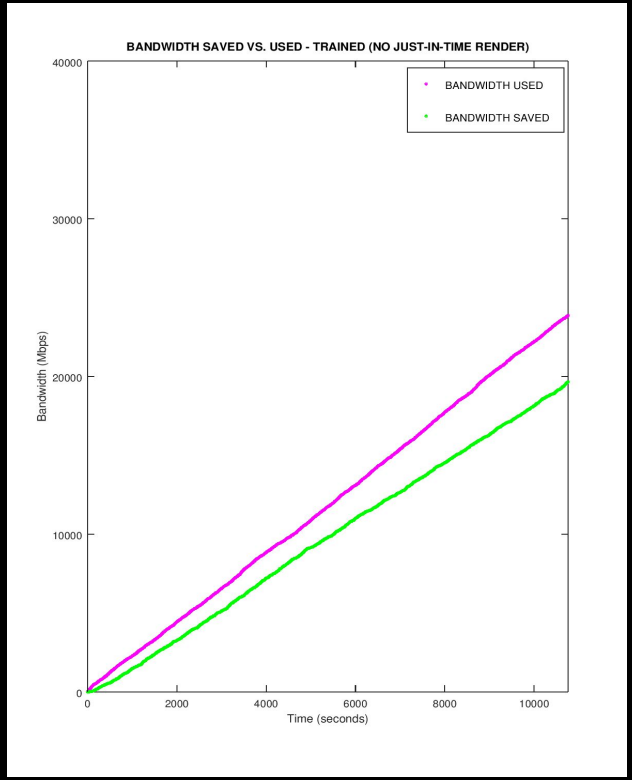
Orange - Today's Smart Caching (Requires Title, Platform, and BitRates)

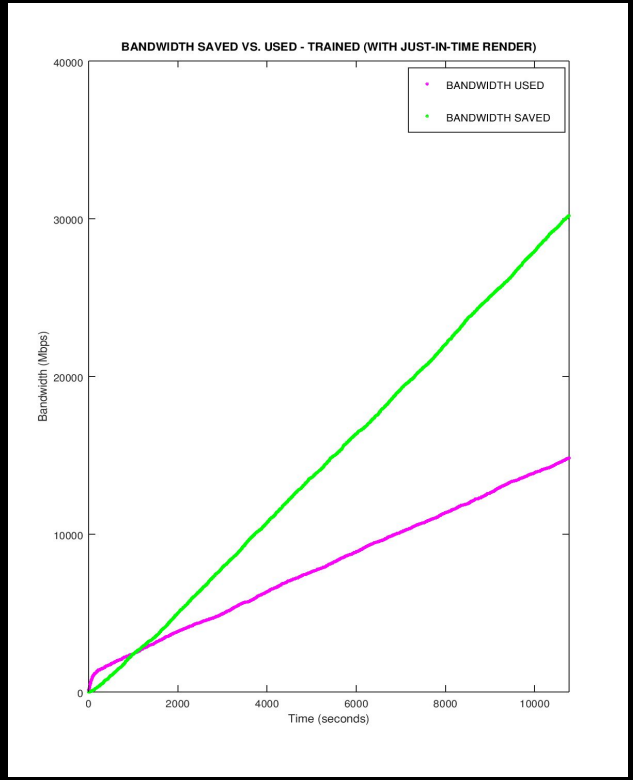
Green - Traditional Caching (fifo, Requires Title, Platform and Bitrates



High Bandwidth Savings vs Bandwidth Used







TRUSTLESS SECURITY MODEL

TRUST & TRUSTLESS DESIGN

- In today's 'client-server' extended architecture
  - We have to trust servers, file systems and databases with our data
  - Storage can be encrypted - but the 'vendors' manage the keys
  - We have to trust the vendors, their vendors, and their infrastructure
- The Content Internet should be "trustless"
  - Trust system must be 'open'
  - All claims are verifiable
  - Crypto framework ensures privacy and integrity
- Commerce and Value should be integrated

BLOCKCHAIN CONTROLLED CONTENT ACCESS

- All content transactions are secured by a transaction on a blockchain ledger
- Trustless design - all claims are verifiable
- Uses a decentralized consensus
- Protected by a trustless crypto framework and protocol for privacy
- Public Audit Trail with Privacy
  - All transactions are publicly available in the distributed ledger
  - Actual information is private (ZERO KNOWLEDGE)
- Private spaces / Trusted spaces
- Versatile, programmable Smart Contracts implement any business value exchange

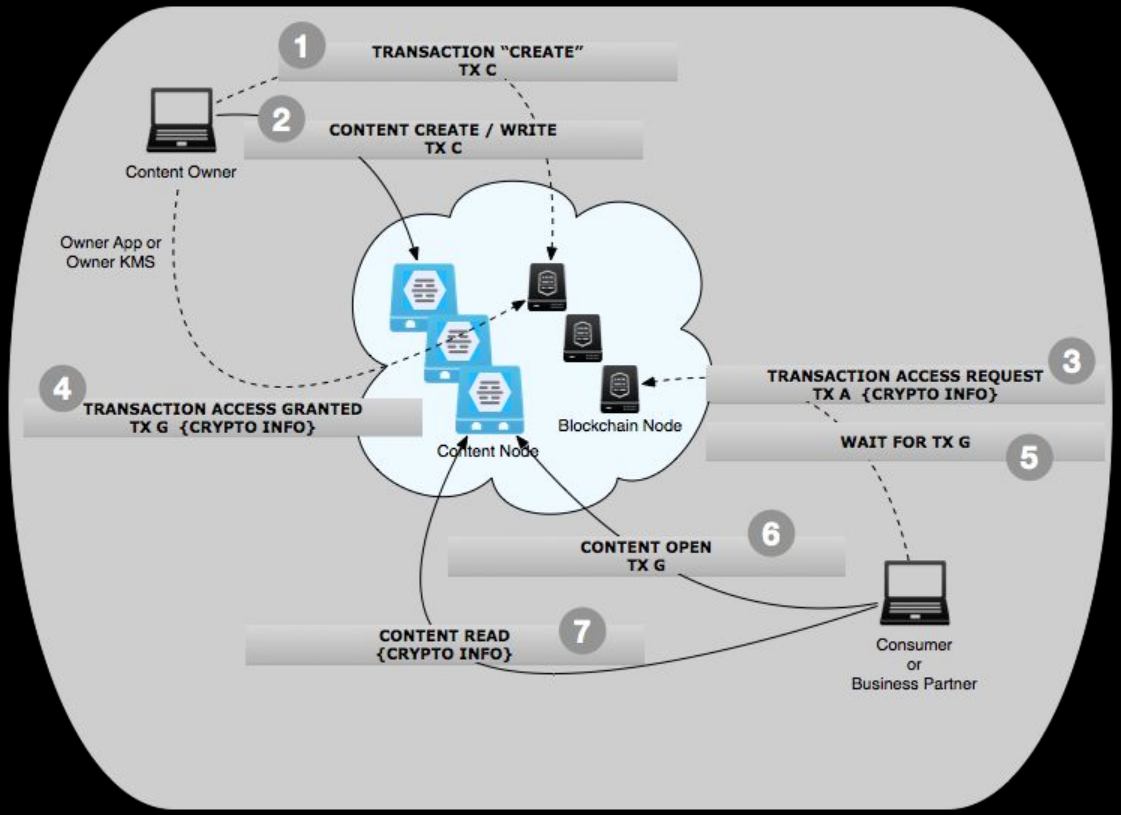
CRYPTO FRAMEWORK

Content is encrypted end-to-end and content secret keys are re-encrypted using proxy re-encryption.

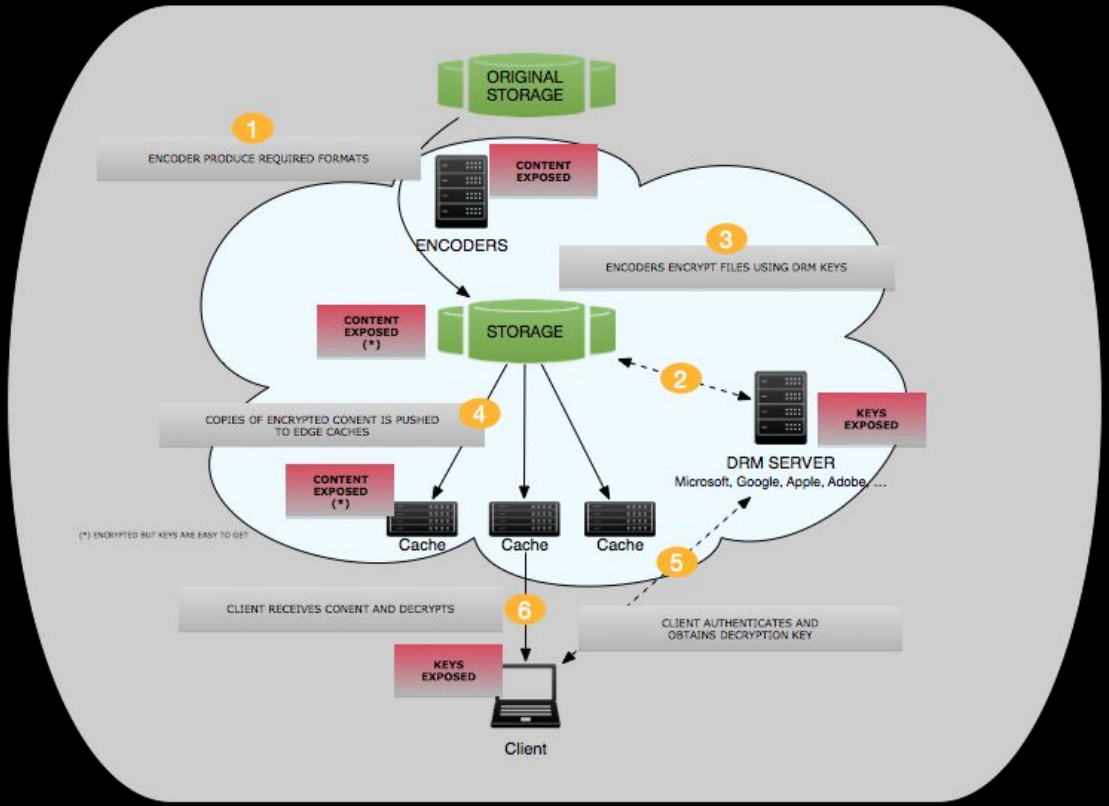
Owner(s) and owner KMS are in charge of secret keys for given content

All write/read operations require a Tx ID. Tx receipts include crypto info only useful to the owner and requestor(s)

Encryption keys are not exposed
Content plain-text is not exposed to any parties other than approved requestors



EG DRM TODAY → A LOT OF EXPOSURE



EXAMPLE PROGRAMMABLE SMART CONTRACT

```
pragma solidity ^0.4.16;

// This contract keeps track of how many times a partner has accessed a content,
// and charges a configurable amount (of tokens) when the client has surpassed
// its allotment.

contract CustomContract {

    struct PartnerObject {
        address id;
        uint count; // number of requests

        // internal elements
        bool valid; // to distinguish non-existing objects
        uint256 keyIndex; // index in the 'key list'
    }
    struct KeyElement {
        address key;
        bool valid; // set to 'false' when the element is deleted
    }
    address creator;
    function () public payable {
    }

    mapping( address => PartnerObject ) partnerObjects; // keymap
    KeyElement[] partnerObjectKeys; // key list; required to iterate

    event PartnerObjectCreate(address id);
    event DbgString(string s);
    event DbgAddress(address a);
    event DbgUint256(uint256 u);
    event DbgUint(uint u);

    function CustomContract() public payable {
        creator = msg.sender;
    }

    function partnerCreate( ) private returns (address id) {
        var o = PartnerObject( msg.sender, 0, true, 0 );
        partnerObjects[o.id] = o;
        o.keyIndex = partnerObjectKeys.push(KeyElement(o.id, true));
        PartnerObjectCreate(o.id);
        return o.id;
    }
```

```
    function getPartnerCount() public constant returns (uint) {
        var o = partnerObjects [ msg.sender ];
        if ( o.id != 0 )
            return o.count;
        else
            return 0;
    }

    function runAccessPre(address sender, uint256 charge, uint256 value,
        bytes customKeys, bytes customValues, bytes signature)
        public payable returns(uint) {
        uint result = 0;
        var o = partnerObjects [ sender ];
        if ( o.id == 0 ) {
            partnerCreate();
        }
        else {
            DbgString("Partner exists:");
            DbgAddress(o.id);
        }
        if(o.count >= 2)    {
            if (value < charge)
                DbgString("Value insufficient. value: charge:");
                DbgUint256(value);
                DbgUint256(charge);
        }
        o.count = o.count + 1;
        result = 1;
        return result;
    }

    function runAccessPost() public returns(uint) {
      uint result = 1;
      return result;
    }

}
```

ETHEREUM BLOCK


```
{
  difficulty: 1,
  extraData: "0xd783010803846765746887676f312e  …  382e31856c696e757800000000000000000c23219",
  gasLimit: 4712388,
  gasUsed: 104966,
  hash: "0x1bbb0764af01d1b819b0e52b0d483a730e6bacef6a40700dbcb723e16309a4ee",
  logsBloom: "0x00000000000000000000000800000  …  00000800000000000000000",
  miner: "0x0000000000000000000000000000000000000000",
  mixHash: "0x0000000000000000000000000000000000000000000000000000000000000000",
  nonce: "0x0000000000000000",
  number: 5129,
  parentHash: "0x5c8c7fdb00993c181f9c2e00b90272f3632f487465d1714922a81c588d9413be",
  receiptsRoot: "0x55071ed2e6a29273009f09be5c828313302549aa18a199678ab821b51a780e76",
  sha3Uncles: "0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347",
  size: 2042,
  stateRoot: "0x9ac77a57b13e8b2c07aa39a2719fbbd1ce0a5d1b2c016ea9ca1c78f7f34761f7",
  timestamp: 1521233520,
  totalDifficulty: 6840,
  transactions: ["0x491eec1b437cb1eae6909e01e9ccf5358d845ae279d0b630c5e1edc99fff9a8c"],
  transactionsRoot: "0xcb2e230a55ea025bff4878b7c8672717fb709a640b0dc0e1b273a39d8d1411a5",
  uncles: []
}
```


"From": Address of account calling the contract

TRANSACTION


```
{
  blockHash: "0x1bbb0764af01d1b819b0e52b0d483a730e6bacef6a40700dbcb723e16309a4ee",
  blockNumber: 5129,
  from: "0x76bbb0984ac61f295a9363bfeb091ae2cfae8e47",
  gas: 1800000,
  gasPrice: 2000000000,
  hash: "0x491eec1b437cb1eae6909e01e9ccf5358d845ae279d0b630c5e1edc99fff9a8c",
  input: "0x24e27684c0bee6ac7dddfa9d9f1048877afa007f0eb  …  127076d0575982afb4ba17f4228d81",
  nonce: 4,
  r: "0x1be4f78e0611fb8a88d22383fdd760ef59980c38efcea6ca53a585a47dd8acbe",
  s: "0x1d6c1da1432753d63604bb831e43acdef25684a2cef3f9a92c55069afacdef5",
  to: "0x1723ec365f6ffb933cc21ee57df8067db0360553",
  transactionIndex: 0,
  v: "0x1c",
  value: 8000000000000000000
}
```


"To": Address of the contract

TRANSACTION RECEIPT

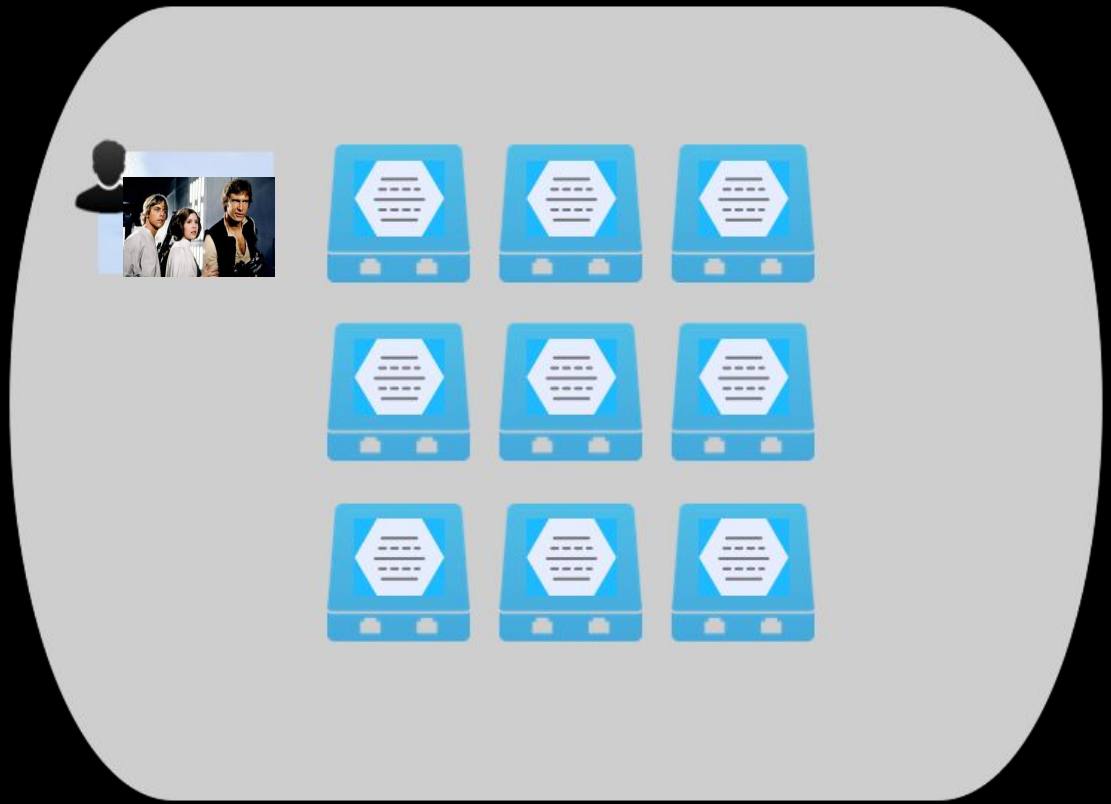

```
{
  blockHash: "0x1bbb0764af01d1b819b0e52b0d483a730e6bacef6a40700dbcb723e16309a4ee",
  blockNumber: 5129,
  contractAddress: null,
  cumulativeGasUsed: 104966,
  from: "0x76bbb0984ac61f295a9363bfeb091ae2cfae8e47",
  gasUsed: 104966,
  logs: [{
     address: "0x1723ec365f6ffb933cc21ee57df8067db0360553",
     blockHash: "0x1bbb0764af01d1b819b0e52b0d483a730e6bacef6a40700dbcb723e16309a4ee",
     blockNumber: 5129,
     data: "0xe6163636573735265717565737 43a0  …  00000000000000000000000000000000",
     logIndex: 0,
     removed: false,
     topics: ["0x415302daeab7bd01c23adb4dee059bc666b313134024f09781aa6d73ff6ec042"],
     transactionHash: "0x491eec1b437cb1eae6909e01e9ccf5358d845ae279d0b630c5e1edc99fff9a8c",
     transactionIndex: 0
  }, {
     address: "0x1723ec365f6ffb933cc21ee57df8067db0360553",
     blockHash: "0x1bbb0764af01d1b819b0e52b0d483a730e6bacef6a40700dbcb723e16309a4ee",
     blockNumber: 5129,
     data: "0x20613535333738233626130303437  …  38636133373332373266616662333839 6232",
     logIndex: 1,
     removed: false,
     topics: ["0x415302daeab7bd01c23adb4dee059bc666b313134024f09781aa6d73ff6ec042"],
     transactionHash: "0x491eec1b437cb1eae6909e01e9ccf5358d845ae279d0b630c5e1edc99fff9a8c",
     transactionIndex: 0
  }, {
     address: "0x1723ec365f6ffb933cc21ee57df8067db0360553",
     blockHash: "0x1bbb0764af01d1b819b0e52b0d483a730e6bacef6a40700dbcb723e16309a4ee",
     blockNumber: 5129,
     data: "0x1152756e437573746f6d507265486f6  …  f6b3a00000000000000000000000000000",
     logIndex: 2,
     removed: false,
     topics: ["0x415302daeab7bd01c23adb4dee059bc666b313134024f09781aa6d73ff6ec042"],
     transactionHash: "0x491eec1b437cb1eae6909e01e9ccf5358d845ae279d0b630c5e1edc99fff9a8c",
     transactionIndex: 0
  }],
  logsBloom: "0x00000000000000000000000000000  …  00000000000000002000000000000000000000",
  root: "0x9ac77a57b13e8b2c07aa39a2719fbbd1ce0a5d1b2c016ea9ca1c78f7f34761f7",
  to: "0x1723ec365f6ffb933cc21ee57df8067db0360553",
  transactionHash: "0x491eec1b437cb1eae6909e01e9ccf5358d845ae279d0b630c5e1edc99fff9a8c",
  transactionIndex: 0
}
```

LIVE DEMOS OF USE CASES

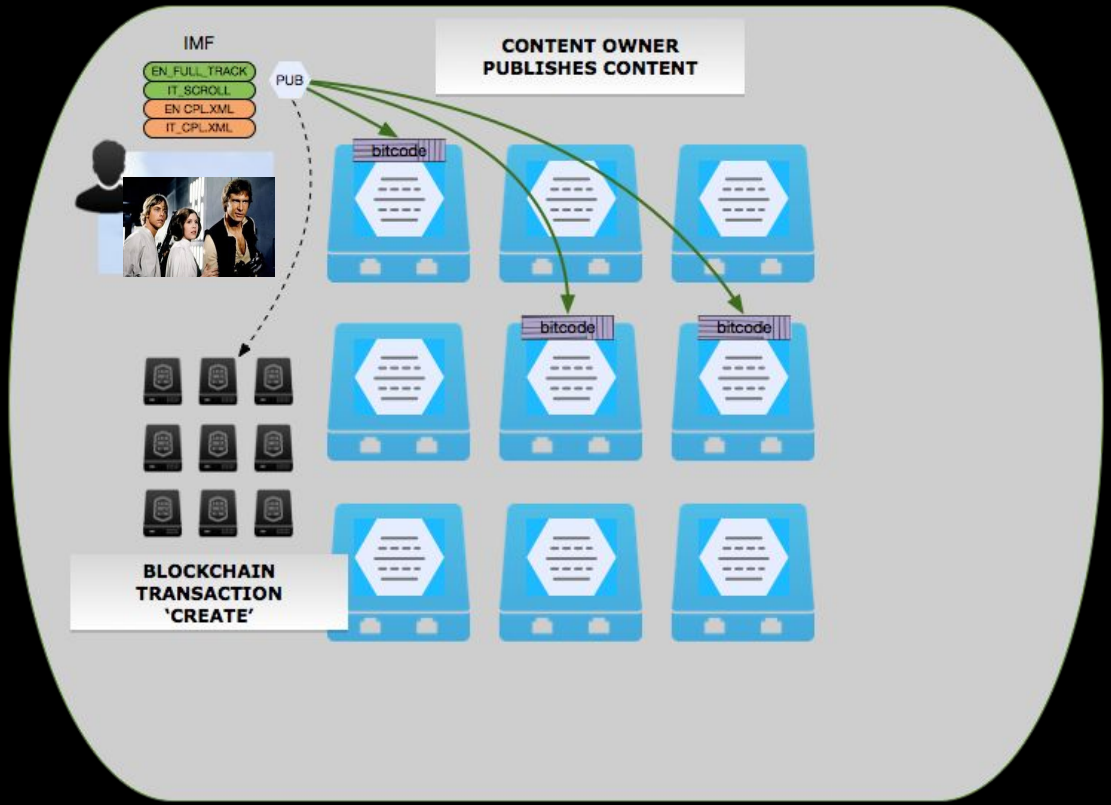
DEMO SCENARIOS

- Global distribution of multi-language/version content from single source
- Instant repair without repackaging, re-transcoding, or redistributing
- Example bitcode ops - ML auto tagging, watermarking, clipping, validation
- Fast content proofs to identify corrupted / invalid / counterfeit content
- Smart Contracts Enabling a Scalable value flow : Consumer / Content Owner / Content Licensee / Sponsor
  - “Free” (paid to watch ad) and “Paid” (ad free) streaming
  - Advertising Marketplace via Bids on Content Tags
  - Electronic Sell Through, e.g. clip, download, with Audience data feedback
  - “Instant” Content licensing and Reconciliation - Accept license agreement, Upload content, Approve content, and Redeem Payment

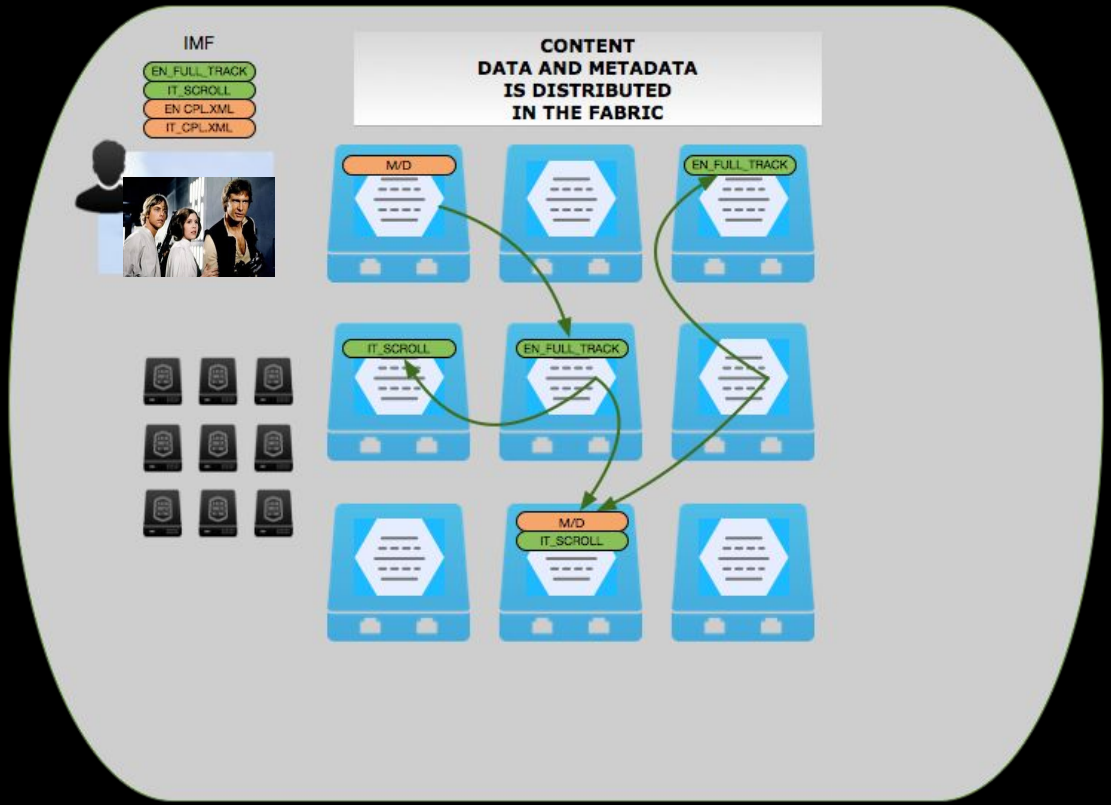
DYNAMIC CONTENT RENDERING FROM IMF



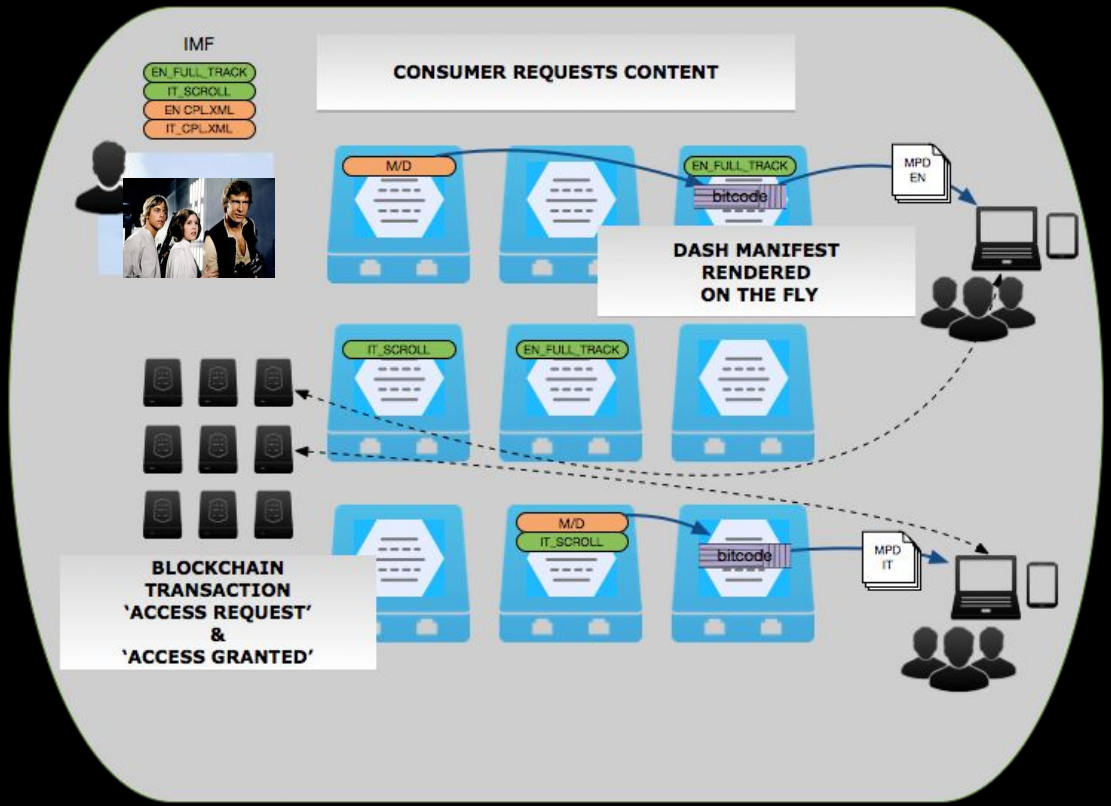
DYNAMIC CONTENT RENDERING FROM IMF



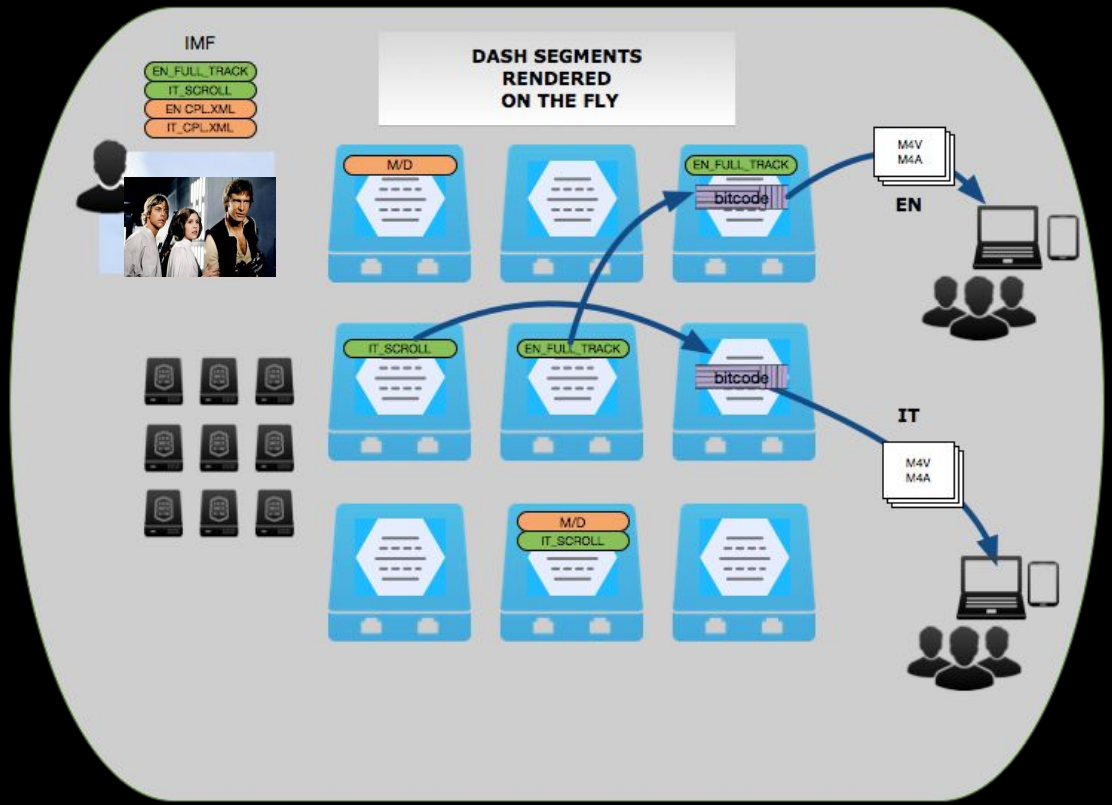
DYNAMIC CONTENT RENDERING WITH IMF



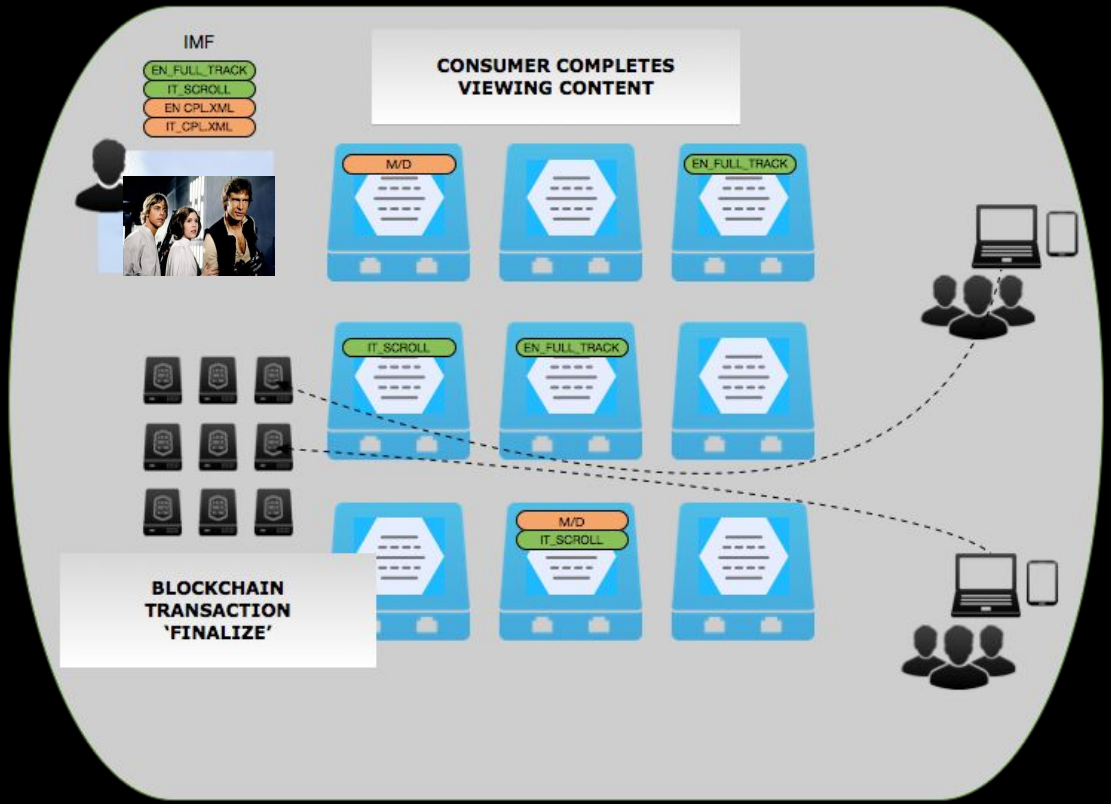
DYNAMIC CONTENT RENDERING WITH IMF



DYNAMIC CONTENT RENDERING WITH IMF



DYNAMIC CONTENT RENDERING WITH IMF



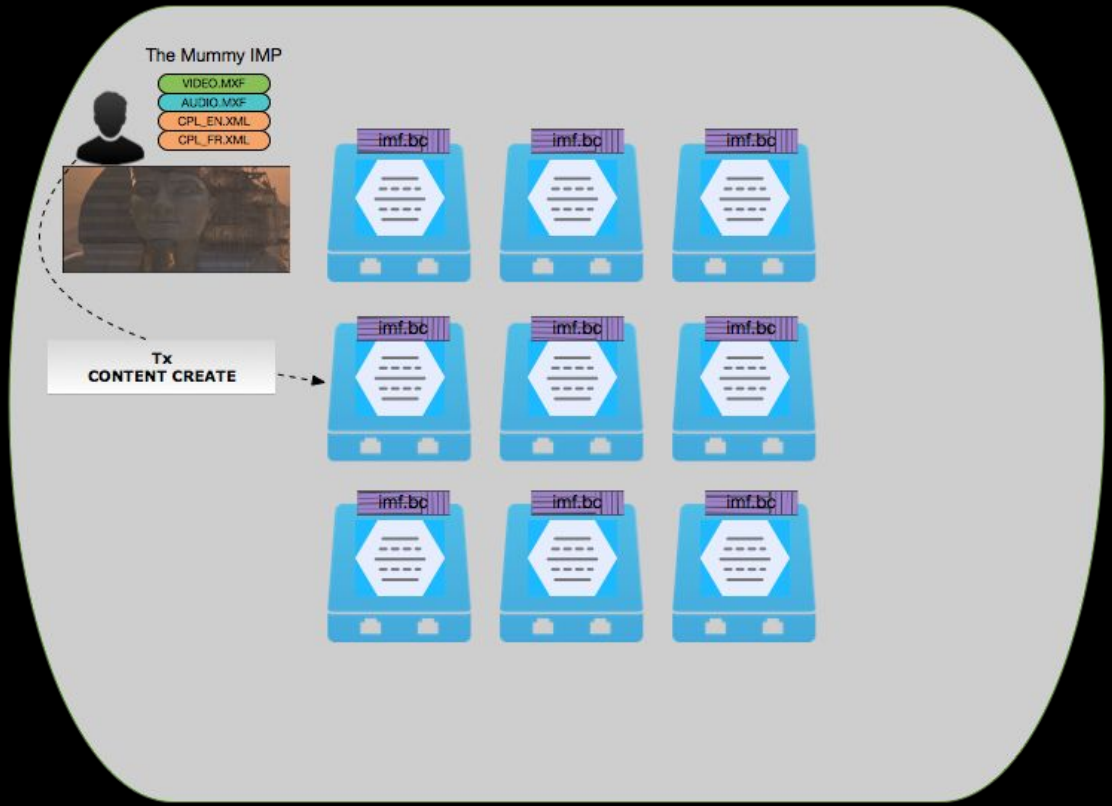
DEMO

FAST REPAIR

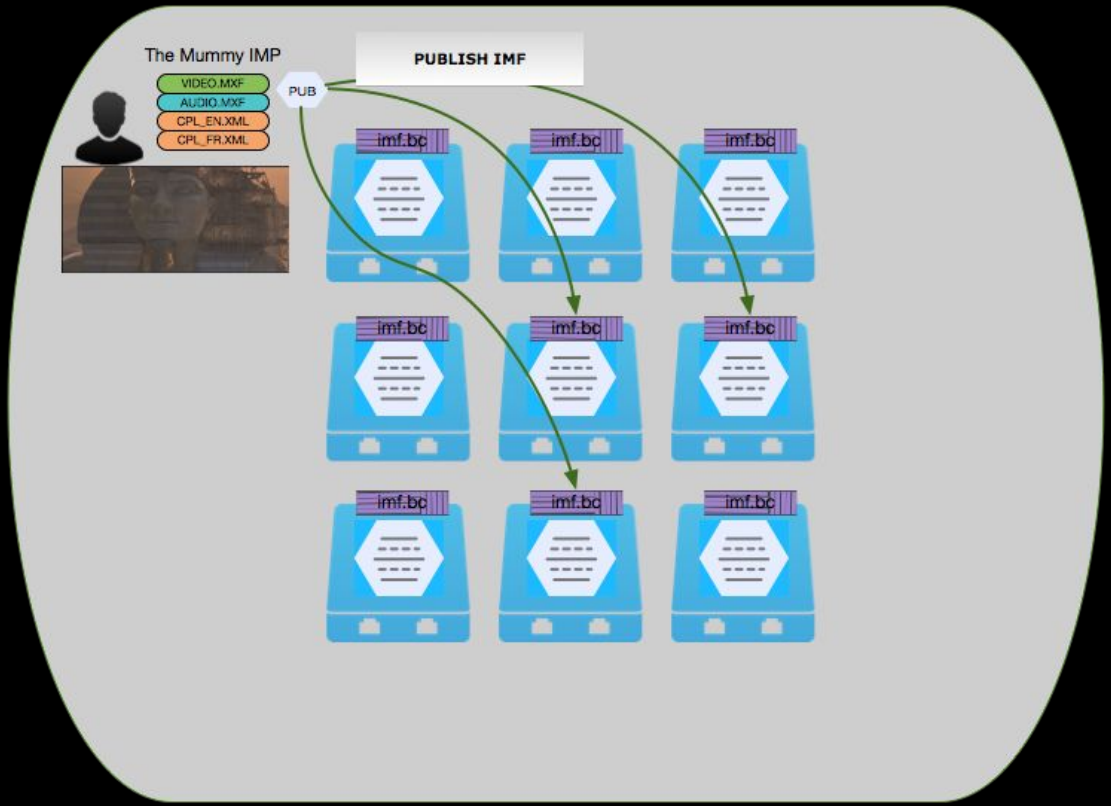
SCENARIO - Multi Version Global Streaming with Instant Repair

- Single IMF Master with multiple language audio tracks
  - English, French, Spanish
  - Content PlayLists (xml) describe geo and language-specific renderings
- Distribute to end users globally using the content fabric
  - (eg as multi-bitrate DASH/HLS)
- Emergency Repair of Missing Audio tracks
  - Instantly, without repackaging or redistributing the original master - fast, flexible, cheap!
- Multi-market bundles
- Maximum quality delivery and Users charged by BW delivered

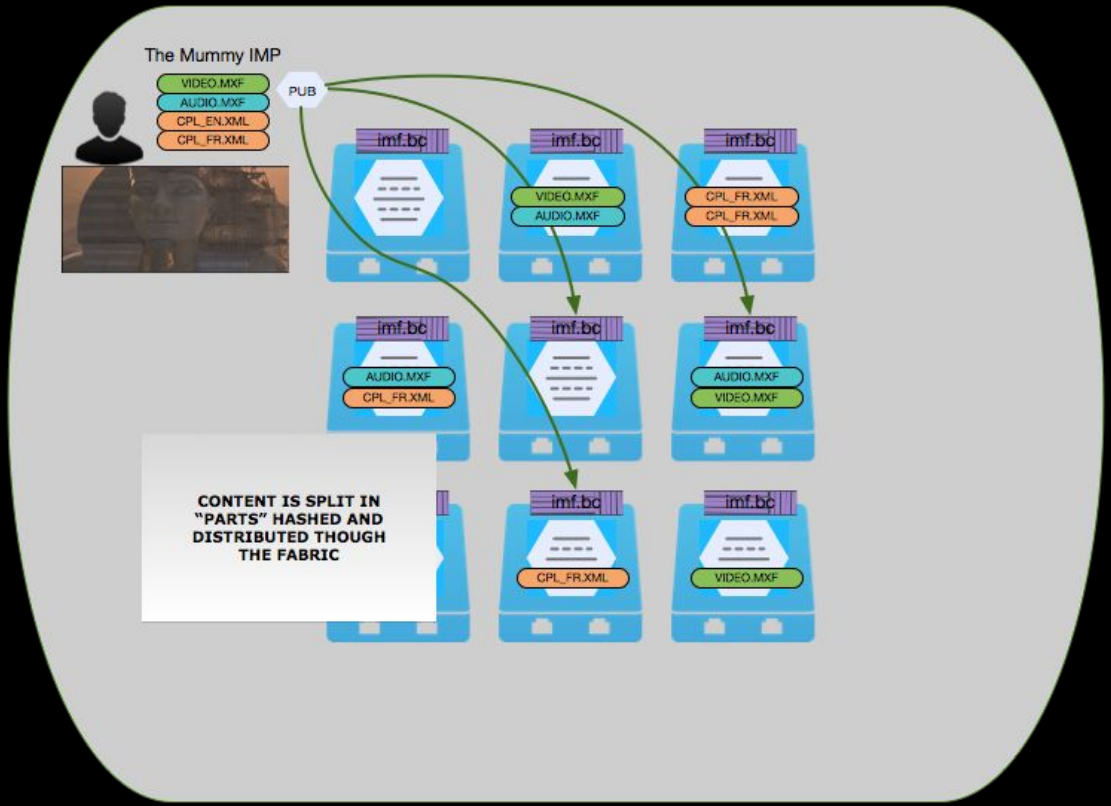
DYNAMIC RENDERING WITH INSTANT REPAIR



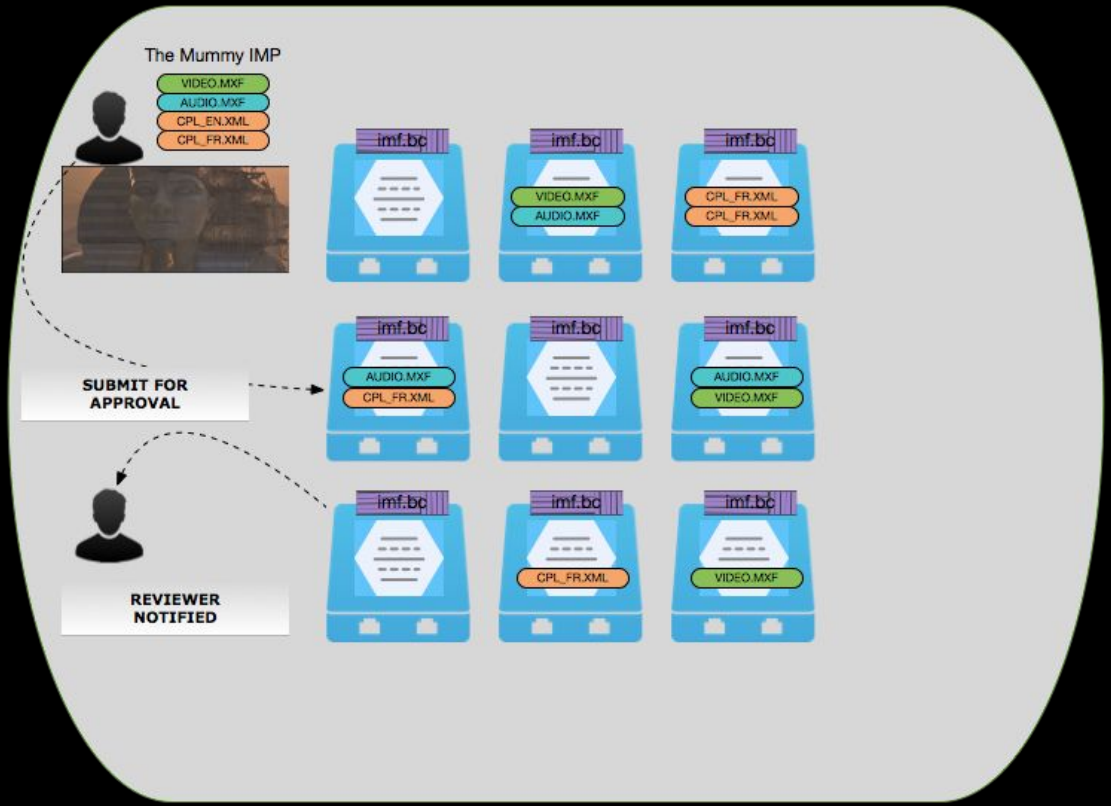
DYNAMIC RENDERING WITH INSTANT REPAIR



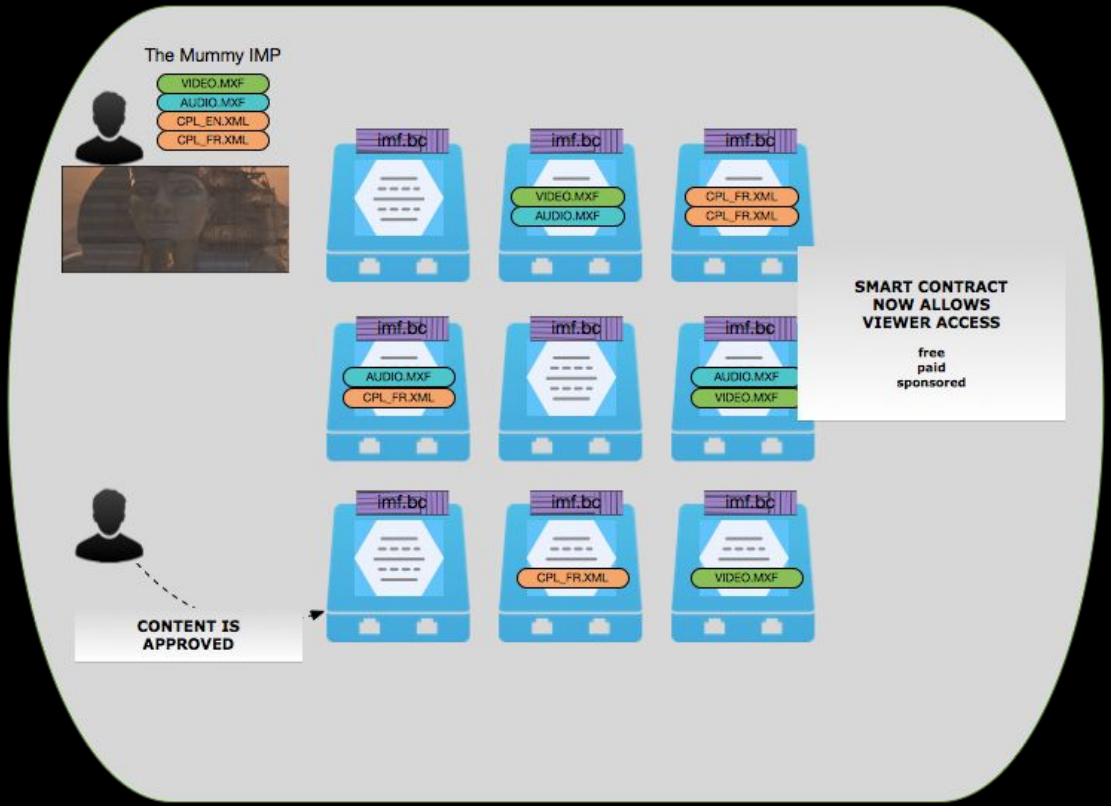
DYNAMIC RENDERING WITH INSTANT REPAIR



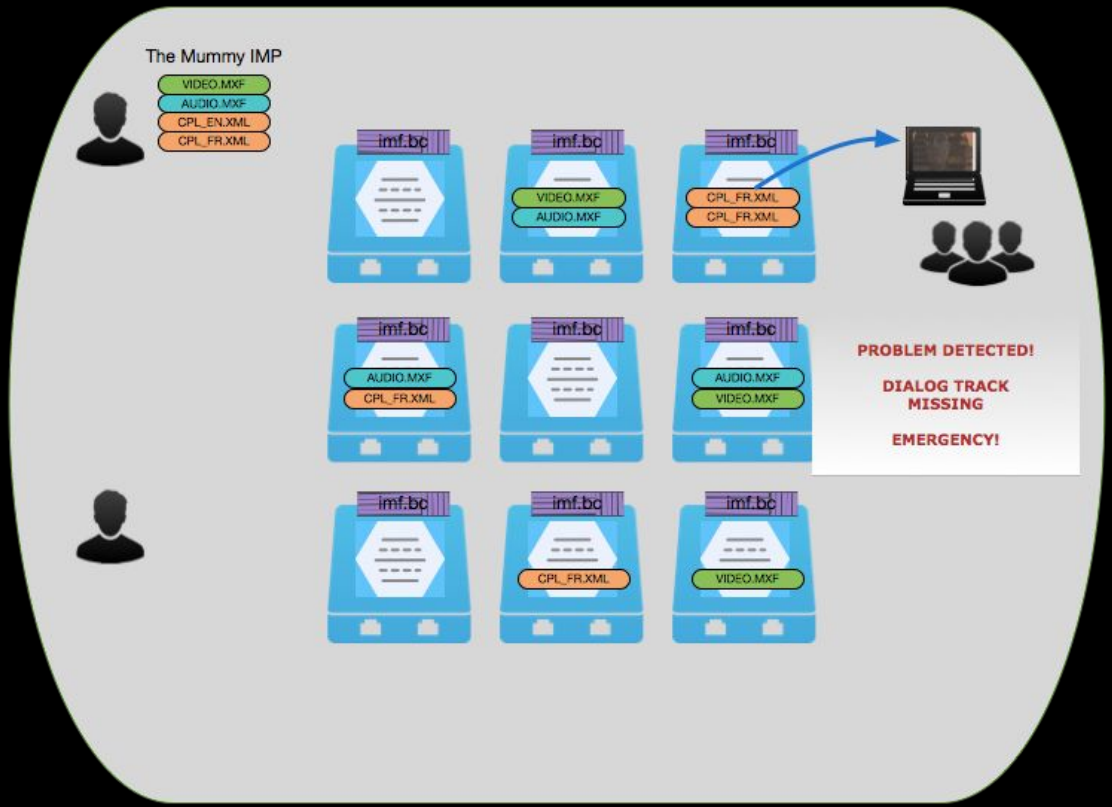
DYNAMIC RENDERING WITH INSTANT REPAIR



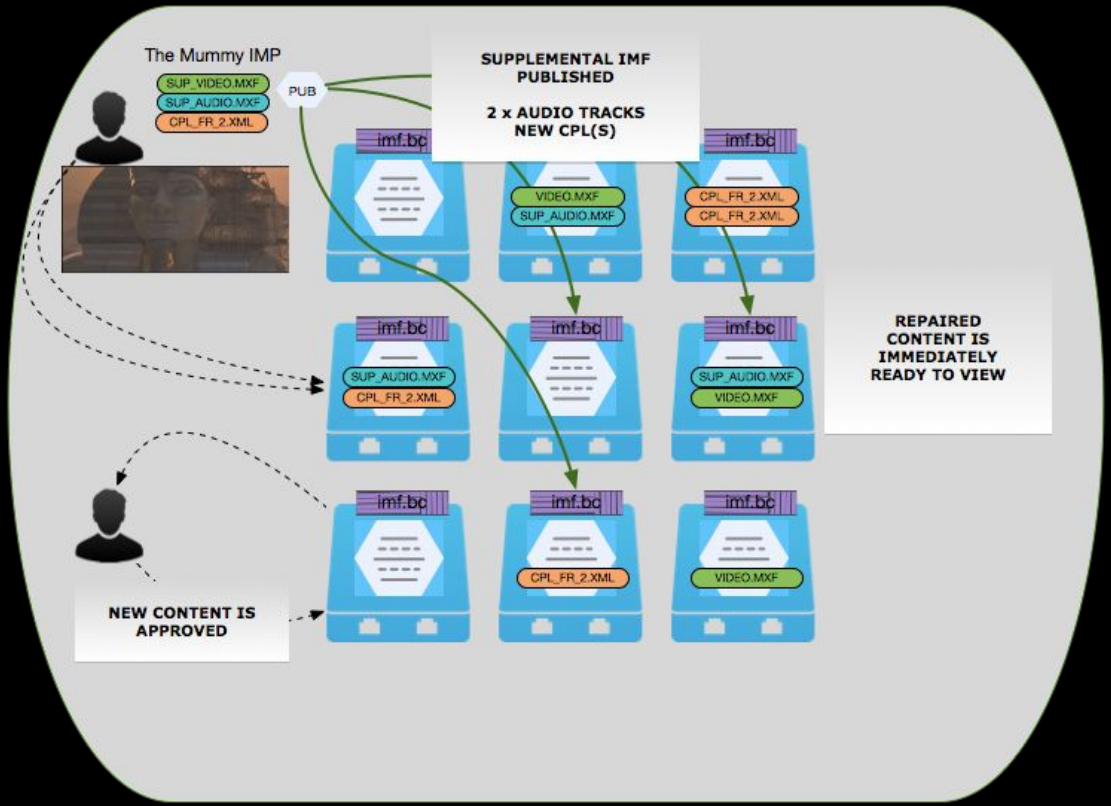
DYNAMIC RENDERING WITH INSTANT REPAIR



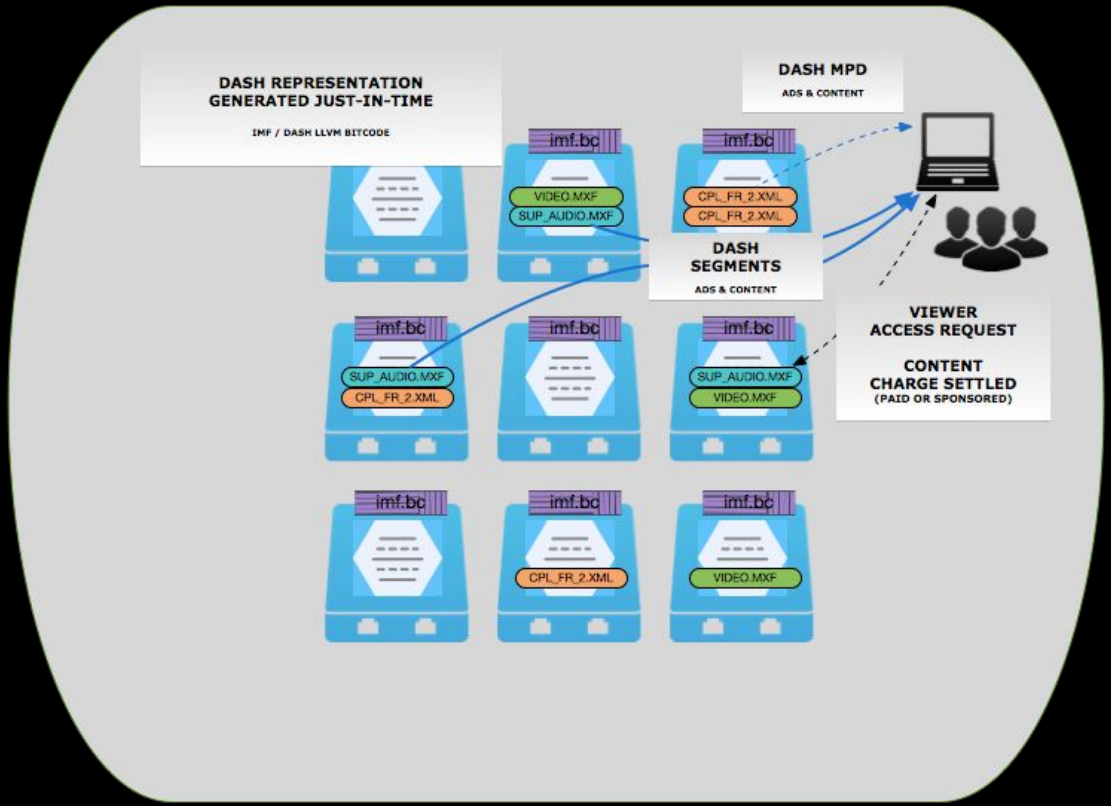
DYNAMIC RENDERING WITH INSTANT REPAIR



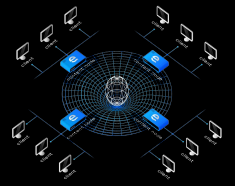
DYNAMIC RENDERING WITH INSTANT REPAIR



DYNAMIC RENDERING WITH INSTANT REPAIR



CONCLUSION



Internet Evolution calls for a new CONTENT FABRIC

- **Comprised of content nodes running a single *software stack***
- Node software is programmable
- Content and metadata stored natively
- Eliminating the need for servers and storage and core BW
- End formats generated just-in-time

Blockchain controlled content publishing and access

- Built on the foundations TRUSTLESS, DECENTRALIZED design
- Native digital content commerce capabilities
- Infinite scale and unbreakable

Potential to Scale the Internet for Content

- Advanced business to business and consumer distribution - including new concepts
- Without servers, storage, databases, workflow engines, asset management or CDNs
- Efficient, secure, and transparent