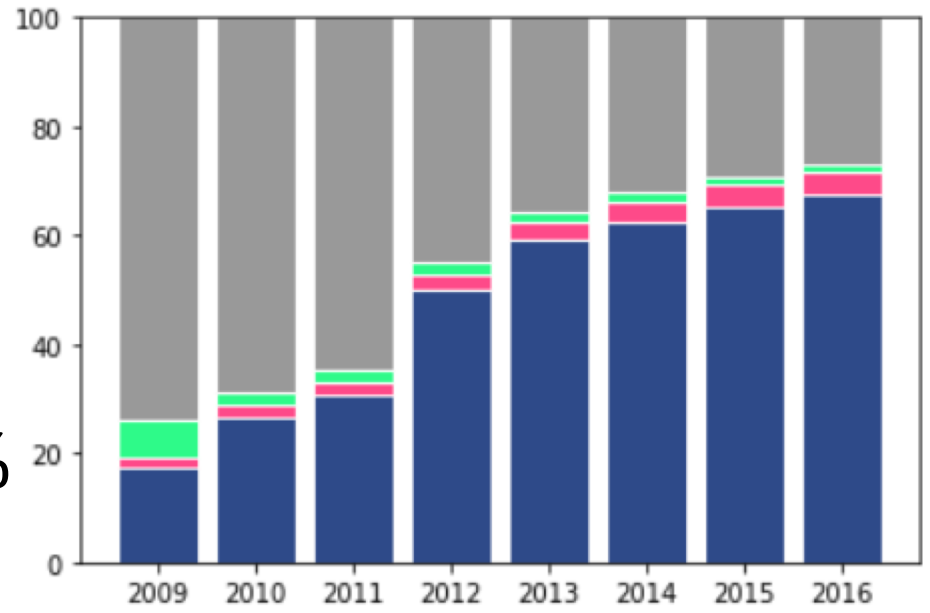
KolmoLD: Data Modelling for the Modern Internet*

Dmitry Borzov, Huawei Canada
Tim Tingqiu Yuan, Huawei
Mikhail Ignatovich, Huawei Canada
Jian Li, Futurewei

*work performed before May 2019

Challenges: Peak Traffic Composition



26%

73%

Video is almost **58%** of the total downstream volume of traffic on the internet

NETFLIX is **15%** of the total downstream volume of traffic across the entire internet

More than **50%** of internet traffic is encrypted, and TLS 1.3 adoption is growing

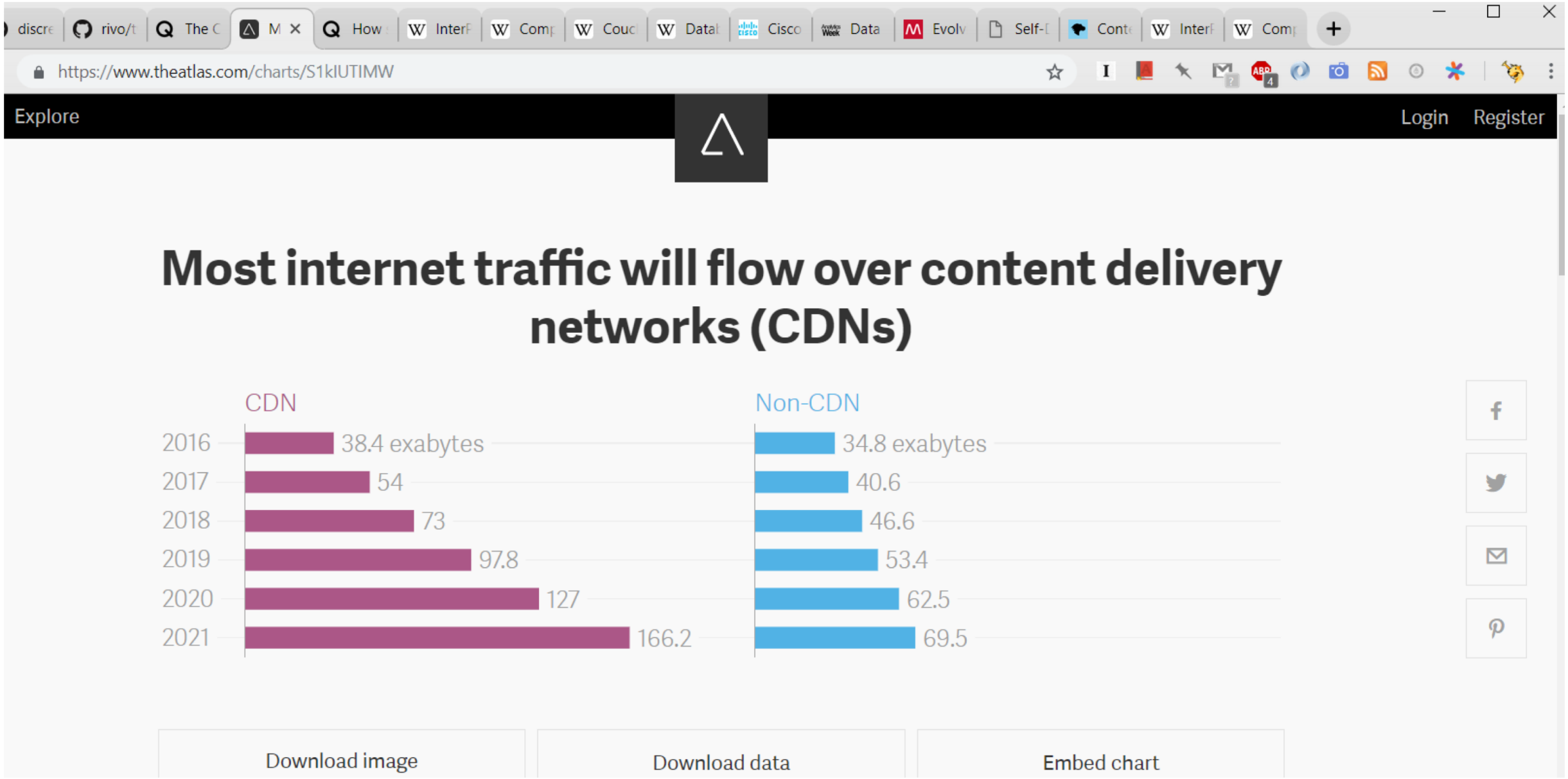
BITTORRENT is almost **22%** of total upstream volume of traffic, and over **31%** in EMEA alone

GAMING is becoming a significant force in traffic volume as gaming downloads, Twitch streaming, and professional gaming go mainstream

Content: sizable, faned-out, static

- Streaming Services (Netflix, Hulu, YouTube, Spotify)
- Software distribution
- File storage services
- Everything else (Instant Messaging, VoIP, Social Media)

[1] Source: Sandvine Global Internet Phenomena reports for 2009, 2010, 2011, 2012, 2013, 2015, 2016, October 2018



[1] https://qz.com/1001569/the-cdn-heavy-internet-in-rich-countries-will-be-unrecognizable-from-the-rest-of-the-worlds-in-five-years/

Technologies to define the revolution of the internet

Founded in 2014
Content-addressable network protocol based on cryptohash naming scheme

Founding company is a YCombinator graduate with backing of high profile SV investors

Founded in 2016
Content-addressable network protocol based on cryptohash naming scheme

Open source project
P2P project
YCombinator graduate

ChunkStream

Video codec
Based on a 2014 MIT research paper

Based on the cryptohash naming scheme

Browser-targeted Runtime

Implemented and supported by all major browsers, an IETF standard

Our Proposal: A data model for interoperable protocols

Content addressing through hashes has become a widely-used means of connecting data in distributed systems, from the blockchains that run your favorite cryptocurrencies, to the commits that back your code, to the web's content at large.

Yet, whilst all of these tools rely on some common primitives, their **specific underlying data structures are not interoperable**.

KolmoLD

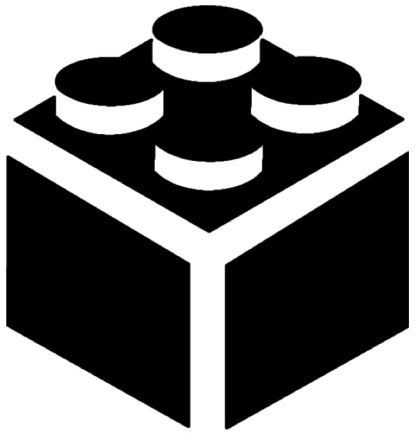
Addressable: connecting layer, inspired by the principles of Kolmogorov complexity theory

Compossible: sending data as code, where code efficiency is theoretically bounded by Kolmogorov complexity

Computable: sandboxed computability by treating data as code

Democratizing networking of generic ICT devices with a principled approach

KolmoLD: Kolmogorov Linked Data



Kolmogorov Content-addressable

Users care about what they want, not who they get it from

Data **composability**

send a way to reproduce data, not data itself

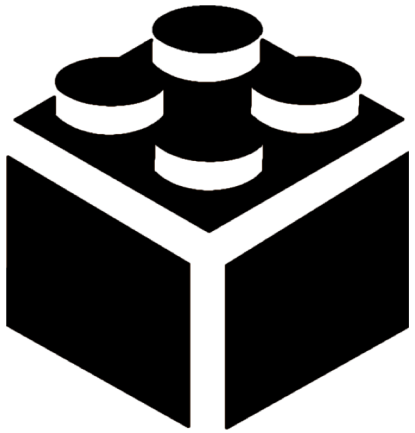
Turing-complete programmability

Sandboxed computability by treating data as code

Content-addressable revolution in the making

	IPFS	Dat Project	ChunkStream	WebAssembly	KolmoLD
Content-addressable cryptohash naming				-	
Composability	-			-	
Turing-complete Programmability	-	-	-		

KolmoLD



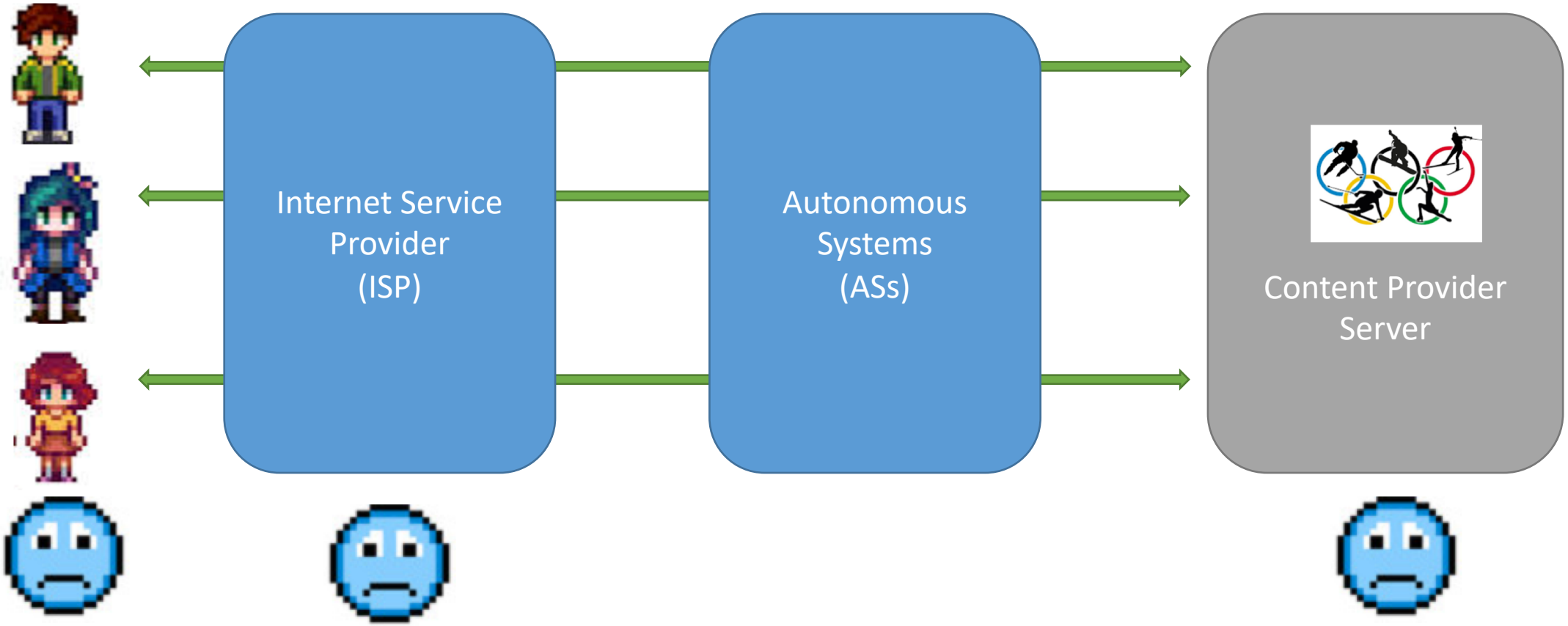
Kolmogorov Content-addressable
Users care about what they want, not who they get it

from

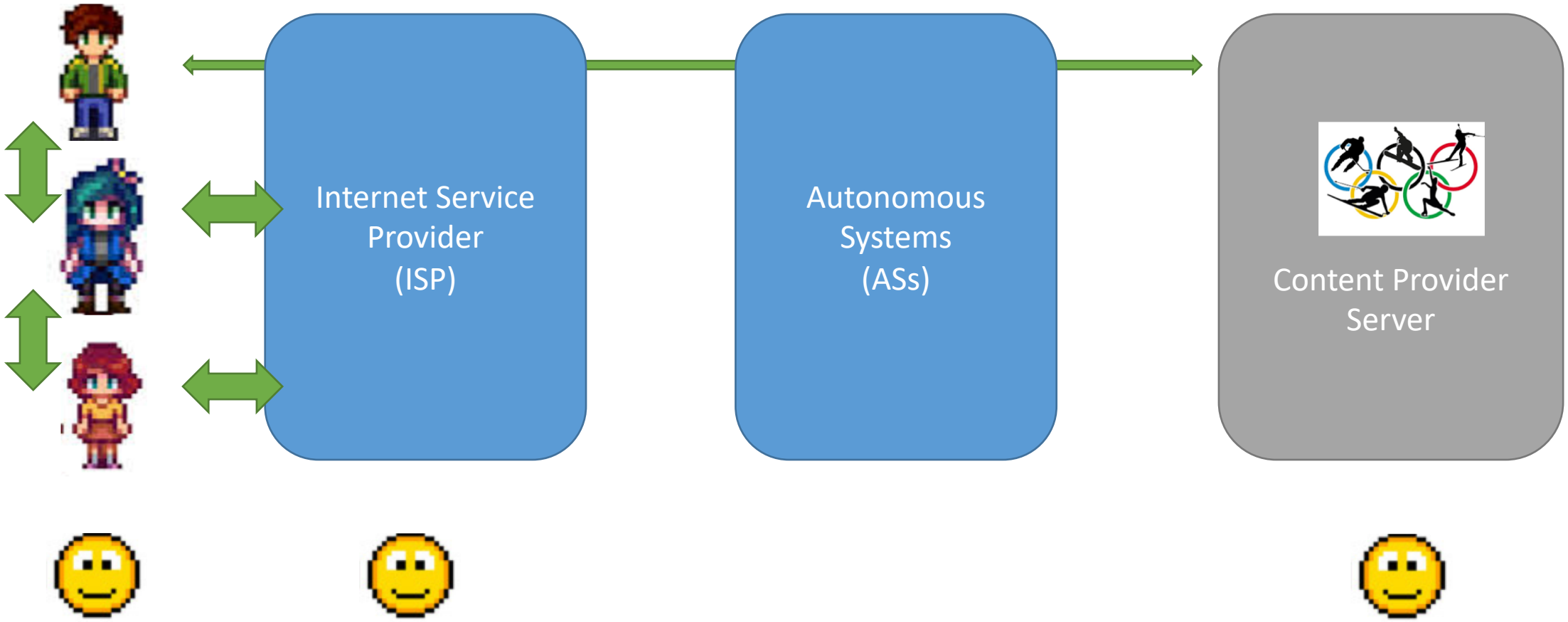
Data composability
send a way to reproduce data, not data itself

Turing-complete programmability
Sandboxed computability by treating data as code

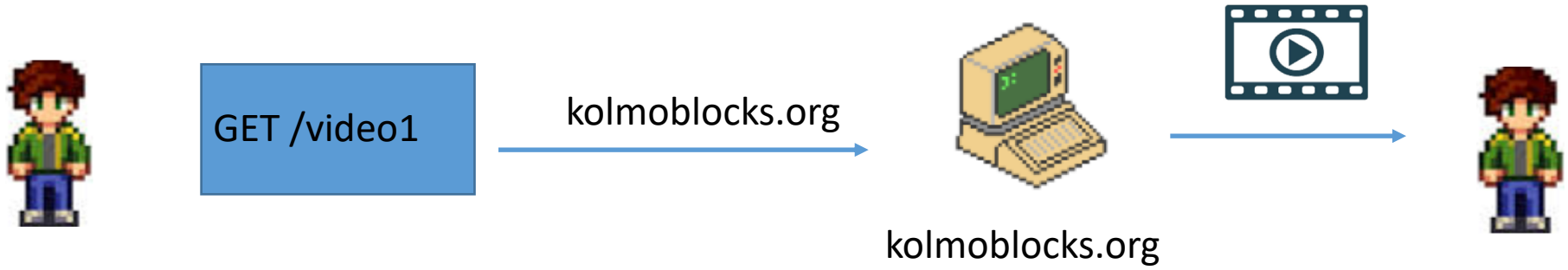
Content-addressable networking: Streaming an Olympics game (Before)



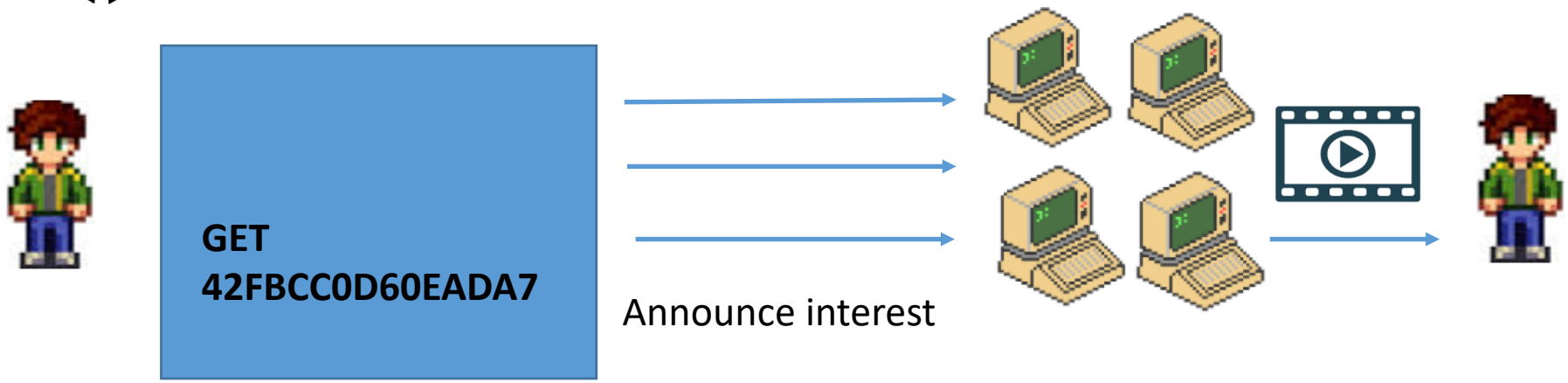
Content-addressable networking: Streaming an Olympics game (After)



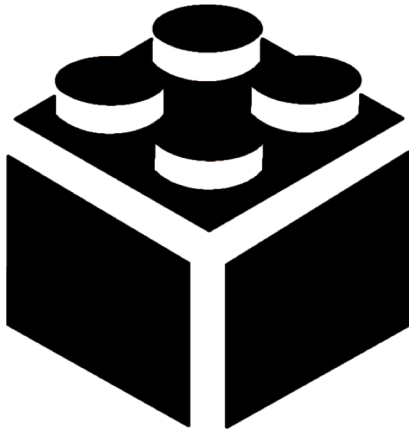
Host-addressable networks



Content-addressable networks

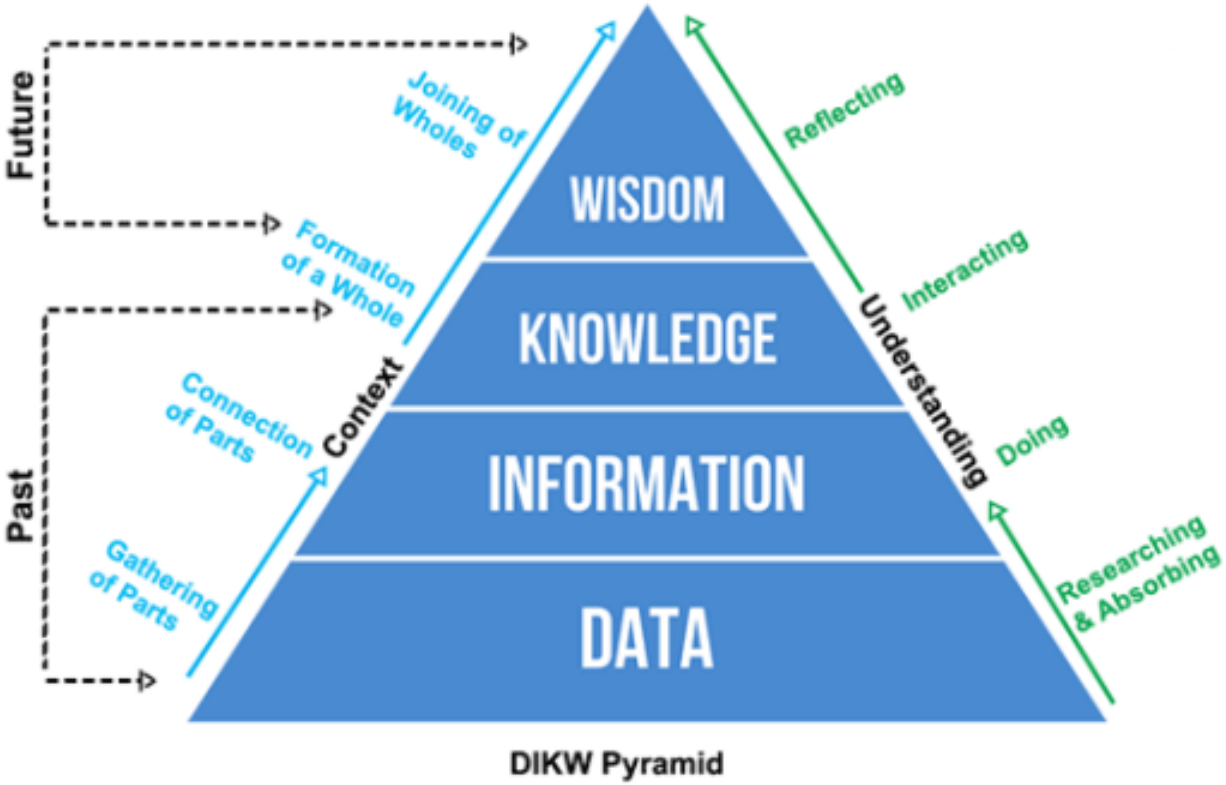


How to represent data content and DIKW in general?

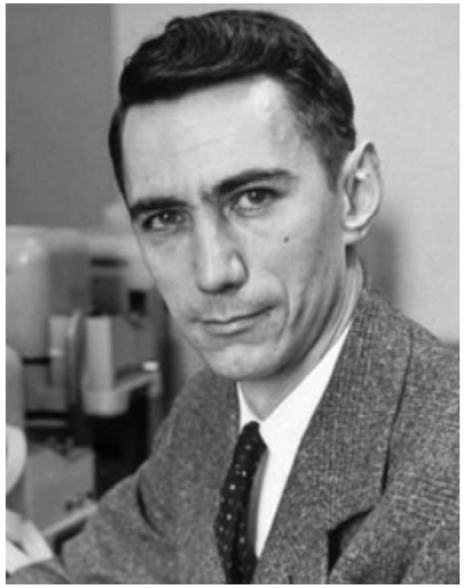


Content-addressable networking

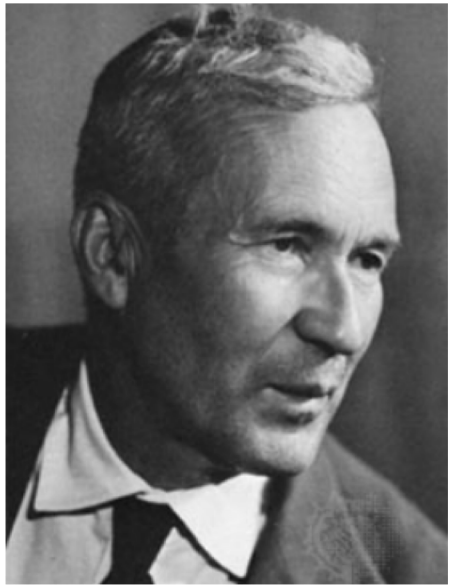
1) Consumers specify what they want, not who they need it from
2) The solution for content distribution



DIKW Pyramid



Claude Shannon
(1916-2001)



Andrei Kolmogorov
(1903-1987)

Alan Turing
(1912-1954)

Kolmogorov complexity:

The shortest unambiguous algorithm, a computer program or code, that will output a given data string.

Shannon entropy = expected [Kolmogorov complexity]

Kolmogorov complexity metric of the given string of data is the size of the shortest algorithm that outputs that data

Kolmogorov complexity https://xkcd.com/1155/

1414213562373095048801688724209698078569671875376948073176

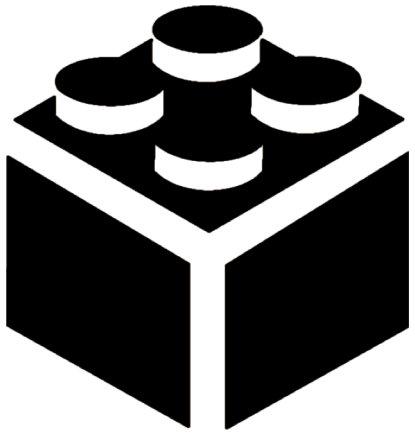
$$\sqrt{2}$$

As an algorithm

Calculate the first 100 60 terms of the series:

$$\sqrt{2} = \sum_{k=0}^{\infty} (-1)^{k+1} \frac{(2k-3)!!}{(2k)!!} = 1 + \frac{1}{2} - \frac{1}{2 \cdot 4} + \frac{1 \cdot 3}{2 \cdot 4 \cdot 6} - \frac{1 \cdot 3 \cdot 5}{2 \cdot 4 \cdot 6 \cdot 8} + \cdots.$$

KolmoLD



Kolmogorov Content-addressable

Users care about what they want, not who they get it from

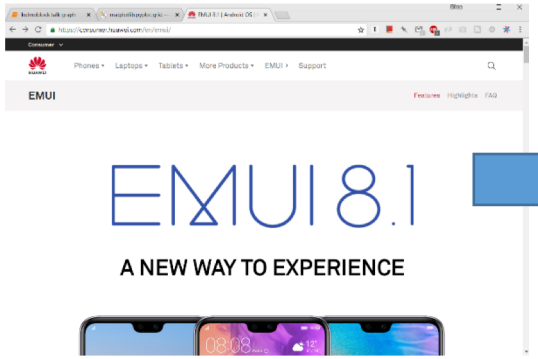
Data composability

send a way to reproduce data, not data itself

Turing-complete programmability

Sandboxed computability by treating data as code

Data composability 1:
Going through the EMUI example



A new version of the OS is released

A 200Mb distro file is distributed across 20M devices across the world

A security patch is issued that flips a single byte at 0x[illegible]24d4588

A patched 200Mb distro file is treated as completely new by the network

Dear phones,
The new version of the distro can be composed out of the original one by flipping the bit at 0xfa24d4588.

Love, the dev team

Data composability 2:
don’t send data, send a way to reproduce data

FD62862

A1EF919

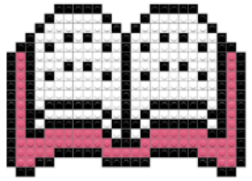
Algorithm

Concatenate images A1EF919, FD62862

Data composability 3: Reusing the encoding table

1Mb, book on potatoes

3Mb, book on cabbages



2Mb, book on tomatoes

Huffman Tree

Huffman-Encoded Text

Huffman Tree

Huffman-Encoded Text

Huffman Tree

Huffman-Encoded Text

$$S_{\mathrm{H}}(N, M) = 27^N + M \cdot \sum_i p(x_i) \log_2 p^{-1}(x_i)$$

$$S = M \cdot \sum_i p'(x_i) \log_2 p^{-1}(x_i)$$

Data Composability 4: Quantifying the impact

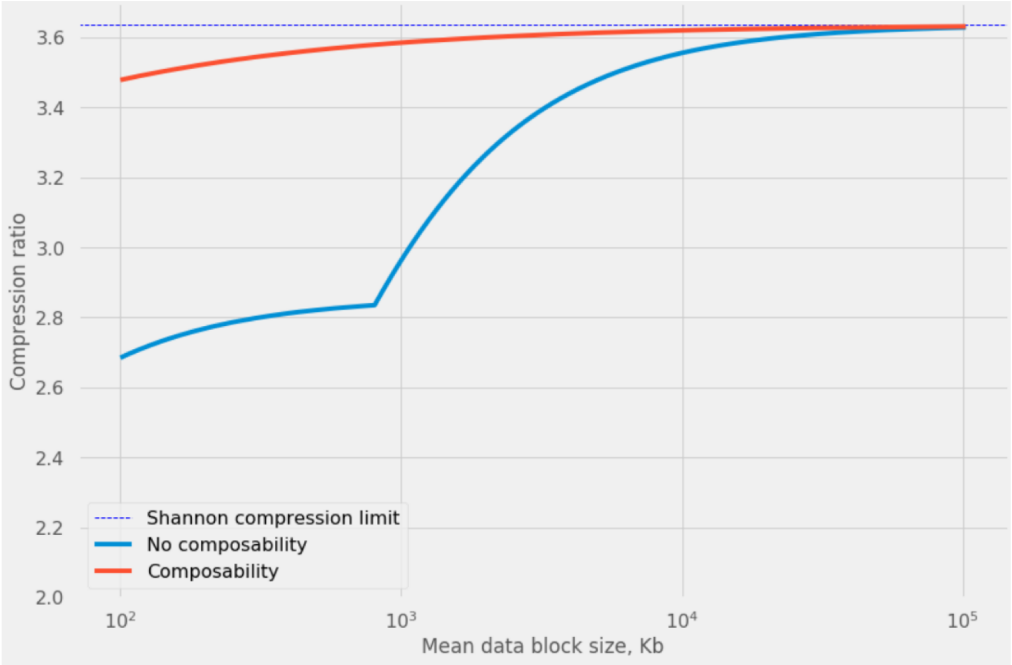
Assumptions:

- Huffman encoding,
- Poisson distribution of the distinctly originating content blocks matching the statistical profile of the reported Internet traffic
- Entropy of the English language text

$$R_H = \frac{M \cdot 8}{27^N + M \cdot G(N)} \qquad R_K(M) = \frac{M \cdot 8}{M \cdot H + \sqrt{M} \cdot \delta L}$$

$$\frac{1}{R_H} - \frac{1}{R_{KH}} = \frac{27^3}{M} - \frac{1}{\sqrt{M}\delta L}$$

Data Composability 4: Quantifying the impact



$$R_H = \frac{M \cdot 8}{27^N + M \cdot G(N)}$$

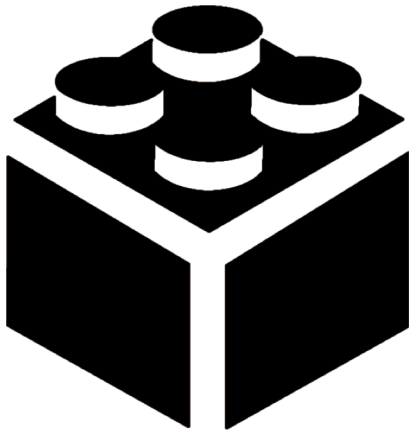
$$R_K(M) = \frac{M \cdot 8}{M \cdot H + \sqrt{M} \cdot \delta L}$$

Data composability

1. Data blocks are identified by cryptographic hash functions
2. Global address space of data
3. Compose data blocks out of other data blocks

You don’t need to send it, just send a way to reproduce the data

KolmoLD



Kolmogorov Content-addressable

Users care about what they want, not who they get it from

Data composability

send a way to reproduce data, not data itself

Turing-complete programmability

Sandboxed computability by treating data as code

Turing-complete programmability 1

Encode data as programs that output given data

ABABABF

```
# SHA256: CC646
def render():
    return “ABABABF”
```

```
# SHA256: CC646
def render():
    return “AB”*3+”F”
```

Turing-complete programmability 2
Referencing other kolmoblocks

ABABABF-FBABABA

SHA256: CC646
ABABABF

```
# SHA256: F3025
def render():
    p = dep("CC646")
    return p+"-"+p[::-1]
```

Turing-complete programmability 3
Reference other data chunks based on cryptohash naming

ABABABF-FBABABA

```
# lambdablock 77650
def f(huffman_tree, encoded_string):
    output = ''
    cur = 0
    while (cur < len(encoded_string)):
        node = huffman_tree
        while type(node) is list:
            bit = encoded_string[cur]
            cur += 1
            node = node[0] if bit else
node[1]
        output += node
    return output
```

```
# kolmoblock
# target block: F3025
# dependancy blocks: 77650
def render():
    huffman_decode = eval(dep('77650'))
    huffman_tree = ['AB',
            ['BA',
                ['F', '-']
            ]
        ]
        encoded = 0b0001101111110101010
        return
huffman_decode(huffman_tree,
encoded)
```

Turing-complete programmability 4
Domain-specific video codecs

	Surveillance video	Self-driving car LIDAR footage	Academic videos (lecture videos, talks and tutorials)
Application-specific requirements	Some objects (people/ car plates) are higher priority	E.g. depth resolution	Text needs to be readable
Probabilistic profile	Static scenery, Traffic & weather	Surrounding traffic video	Emphasis on text, Illustrations, screencasting
Community / engineering resources to support it	Billion dollar industry	Self-driving car industry	Academic & Huawei community

Turing-complete programmability 5
Timelines of adoption of new codecs is in decades

Hello WebAssembly – Co | 3.5 Simple Random Samp | VP8 - Wikipedia

https://en.wikipedia.org/wiki/VP8

Not logged in Talk Contributions Create account Log in

Article Talk | Read Edit View history | Search Wikipedia

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction

Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools

What links here
Related changes
Upload file
Special pages
Permanent link
Page information

VP8

From Wikipedia, the free encyclopedia

For other uses, see VP8 (disambiguation).

VP8 is an open and royalty free video compression format owned by Google and created by On2 Technologies as a successor to VP7.

In May 2010, after the purchase of On2 Technologies, Google provided an irrevocable patent promise on its patents for implementing the VP8 format, and released a specification of the format under the Creative Commons Attribution 3.0 license.[1] That same year, Google also released libvpx, the reference implementation of VP8, under the revised BSD license.[2]

Opera, Firefox, Chrome, and Chromium support playing VP8 video in HTML5 video tag.[3] Internet Explorer officially supports VP8 with a separate codec.[4] According to Google VP8 is mainly used in connection with WebRTC and as a format for short looped animations, as a replacement for the Graphics Interchange Format (GIF).[5]

VP8 can be multiplexed into the Matroska-based container format WebM along with Vorbis and Opus audio. The image format WebP is based on VP8's intra-frame coding. VP8's direct successor, VP9, and the emerging royalty-free internet video format AV1 from the Alliance for Open Media (AOMedia) are based on VP8.[6]

Contents [hide]

1 Features

VP8

VP8

Developed by	Google
Initial release	September 13, 2008
Type of format	Compressed video
Contained by	WebM, Matroska
Extended from	VP7
Extended to	VP9
Standard	RFC6386
Open format?	Yes (specification under CC-by)[1]

Turing-complete programmability 6
Vendoring software

Dynamic libraries
.dll / .so

Go compiler binaries:
Everything is statically included

Containerization:
Docker, Unikernels etc

SHA256: CC646:

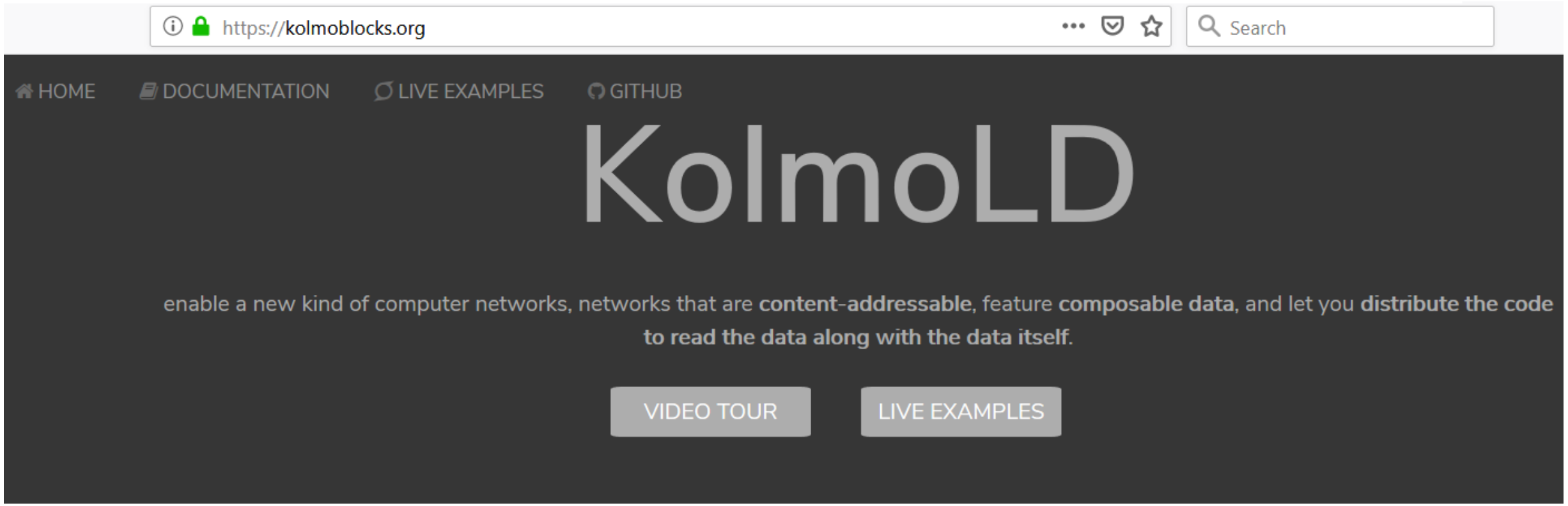
```
# Fibonacci
def fibonacci(n):
  fib = [0] * (n + 1)
  fib[0] = 0  fib[1] = 1
  for i in range(2, n + 1):
    fib[i] = fib[i - 1] + fib[i - 2]  return fib[n]
```

Turing-complete programmability

1. Send data as programs that output the given data
2. Implement a sandboxed runtime that is secure and deterministic
3. Distribute the code along with the data

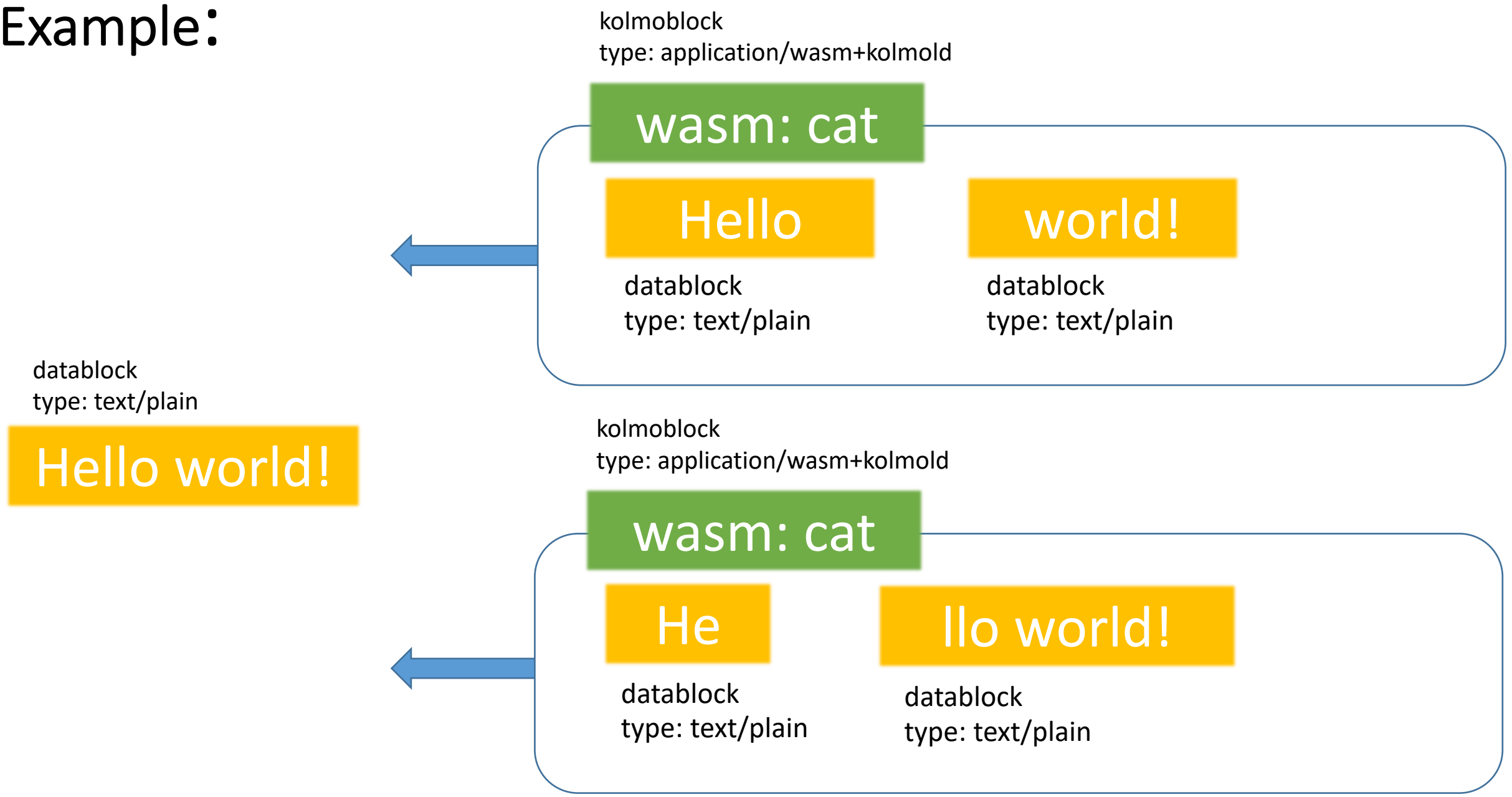
Data as code: distribute the data and the code that reads it along the same channel

More details and live demos @ https://kolmoblocks.org/



KolmoLD: Data Modeling for the Modern Internet

Example:


kolmoblock
type: application/wasm+kolmold
wasm: cat
Hello
world!
datablock
type: text/plain
datablock
type: text/plain
datablock
type: text/plain
Hello world!
kolmoblock
type: application/wasm+kolmold
wasm: cat
He
llo world!
datablock
type: text/plain
datablock
type: text/plain

Example:

```
(match,
  (cid, "B5D40"),
    (exec
        (cid, "cat",
            type: "application/wasm+kolmold"
        ),
        (cid,
            type: "text/plain",
            data: "Hello ",
        )
        (cid,
            type: "text/plain",
            data: "world",
        ),
    )
)

(match,
  (cid, "B5D40"),
    (exec
        (cid, "cat",
            type: "application/wasm+kolmold"
        ),
        (cid,
            type: "text/plain",
            data: "He",
        )
        (cid,
            type: "text/plain",
            data: "llo world",
        ),
    )
)
```

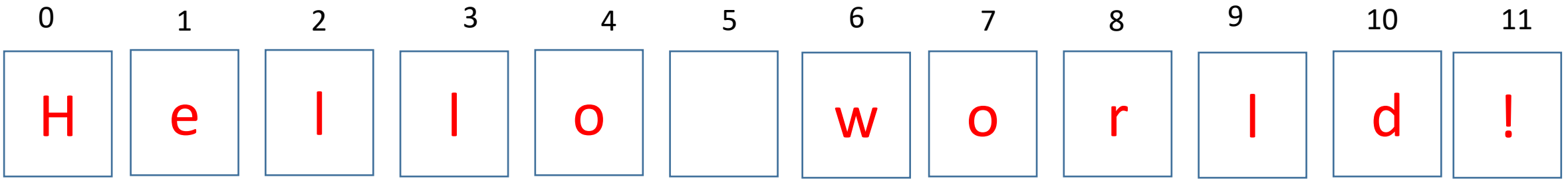
Kolmoblock, cid:"cat"
type: application/wasm+kolmold

```
def main():
    out.low = dep1.low
    out.high = dep2.high
```

Globals of wasm module's instance

dep1.low=0
dep1.high=6
dep2.low=6
dep2.high=12
out.low
out.high

Linear memory of the wasm module's instance



Hello

world!

dep1, dependency datablock 1

dep2, dependency datablock 2

KolmoLD: Kolmogorov Linked Data

Addressable: connecting layer, inspired by the principles of Kolmogorov complexity theory

Compossible: sending data as code, where code efficiency is theoretically bounded by Kolmogorov complexity

Computable: sandboxed computability by treating data as code

Democratizing networking of generic ICT devices with a principled approach

Thanks!

KolmoLD: Data Modelling for the Modern Internet*

Dmitry Borzov, Huawei Canada
Tim Tingqiu Yuan, Huawei
Mikhail Ignatovich, Huawei Canada
Jian Li, Futurewei

*work performed before May 2019