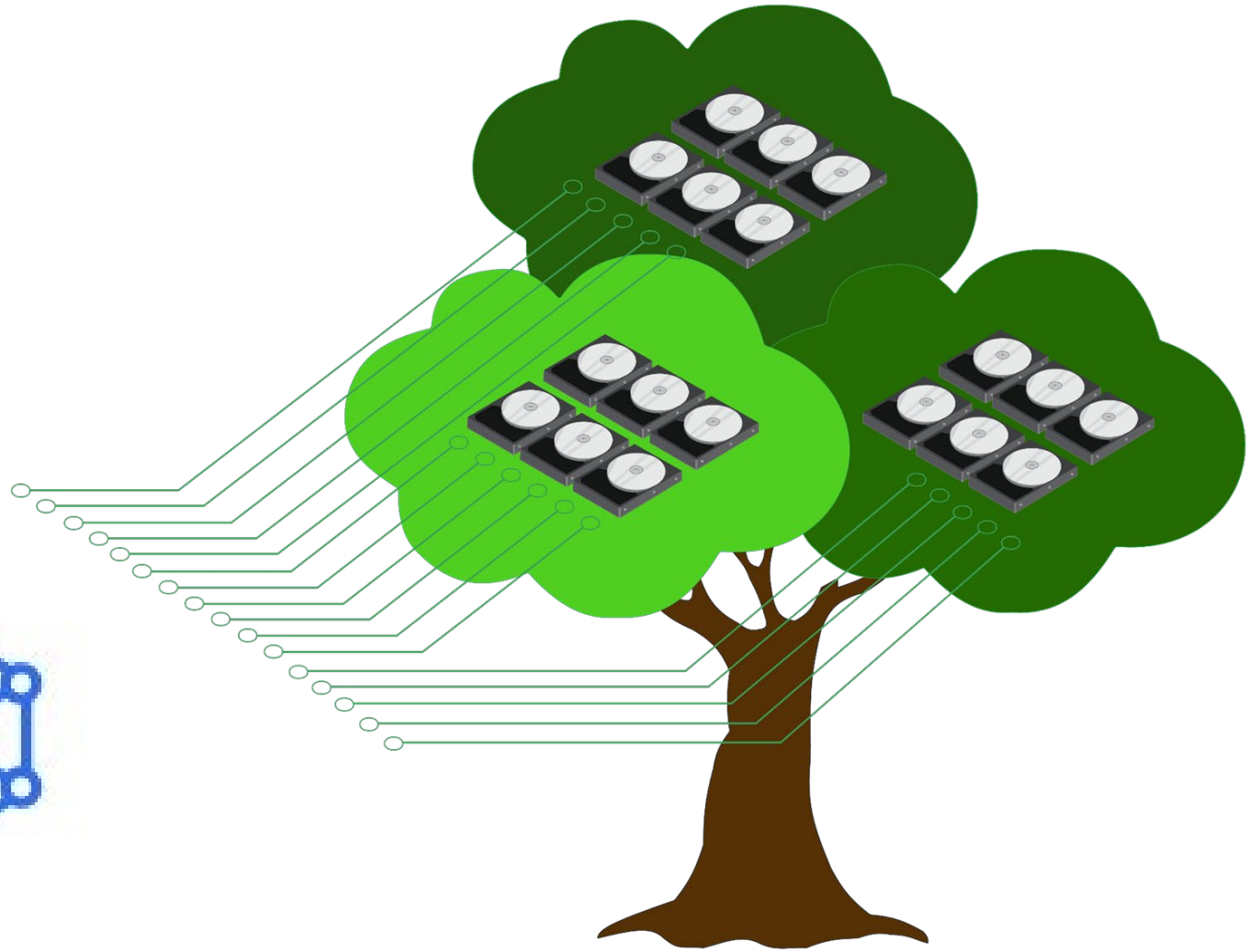
STORAGE DEVELOPER CONFERENCE
SDC 22
Fremont, CA
September 12-15, 2022
BY Developers FOR Developers

A SNIA Event

Lessons Learned (the hard way) from Building a Global, Decentralized Storage Network

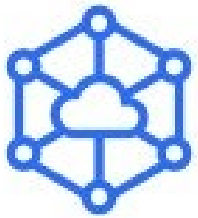
Presented by Dan Willoughby

Principal Engineer, Storj



Agenda

- How Storj Works
- Deeper dive: Erasure coding, Long Tails, Repair
- Texas Blackout 2021
- Lessons learned the hard way
- Questions



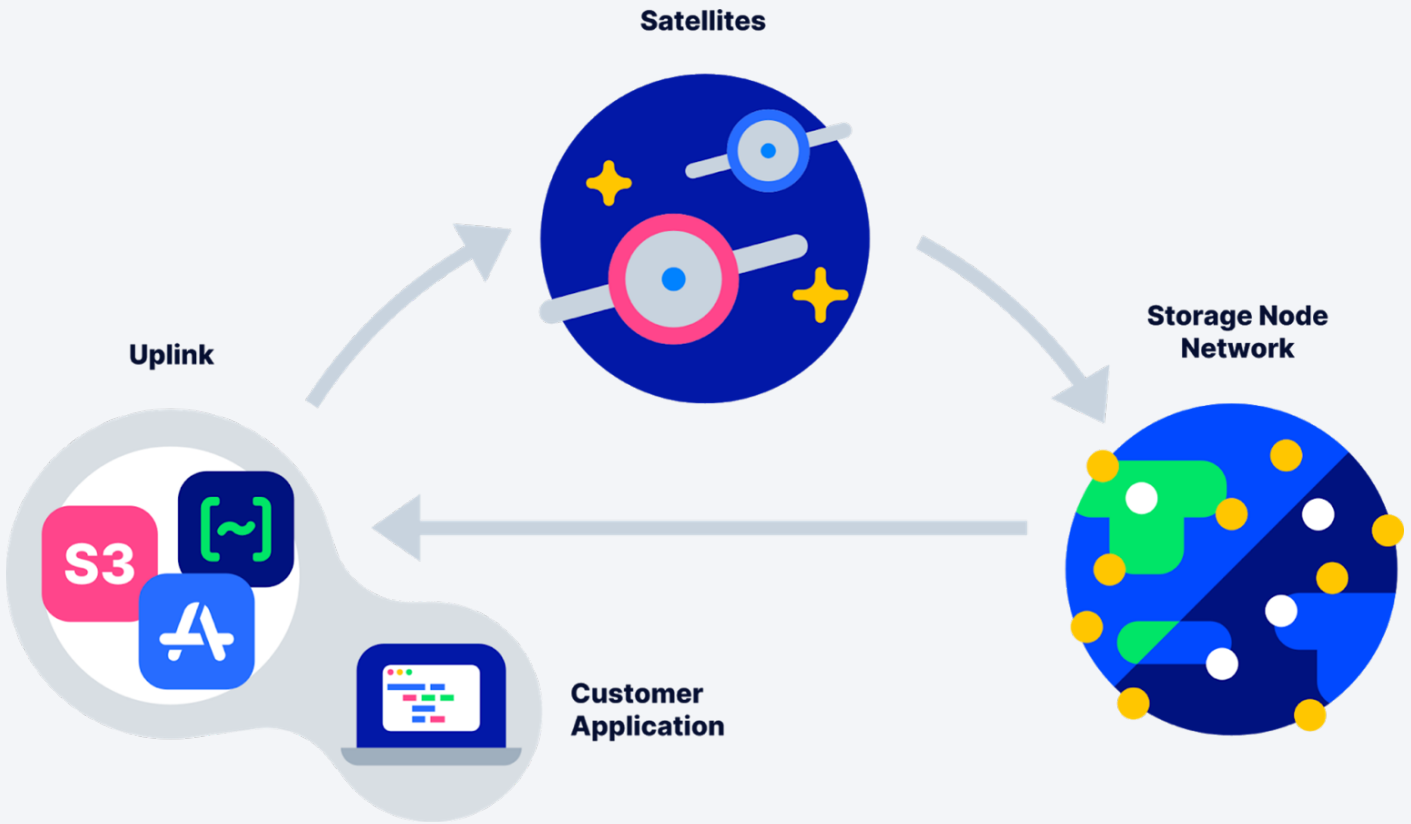
Dan Willoughby

Principal Engineer @ Storj

@plainice_

THE UNIVERSITY OF UTAH

How It Works



Storj Nodes — Supply

Thousands of shared hard drives store pieces of data on the network, without access to any complete file or usable data. Node operators fairly (and profitably) compensated.

Applications — Demand

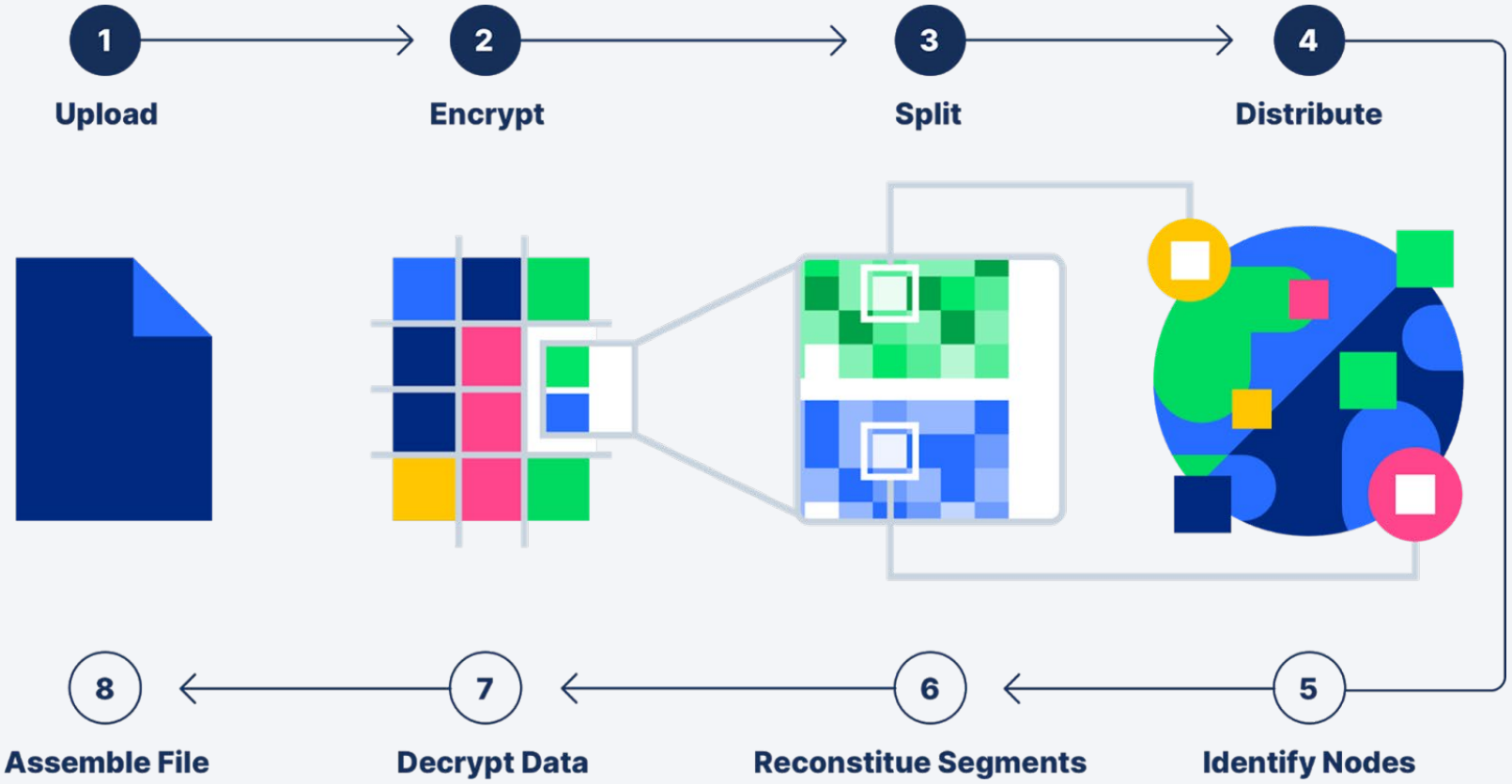
Client applications store encrypted and encoded files split into fragments and stored across the distributed storage network.

Satellites — Network

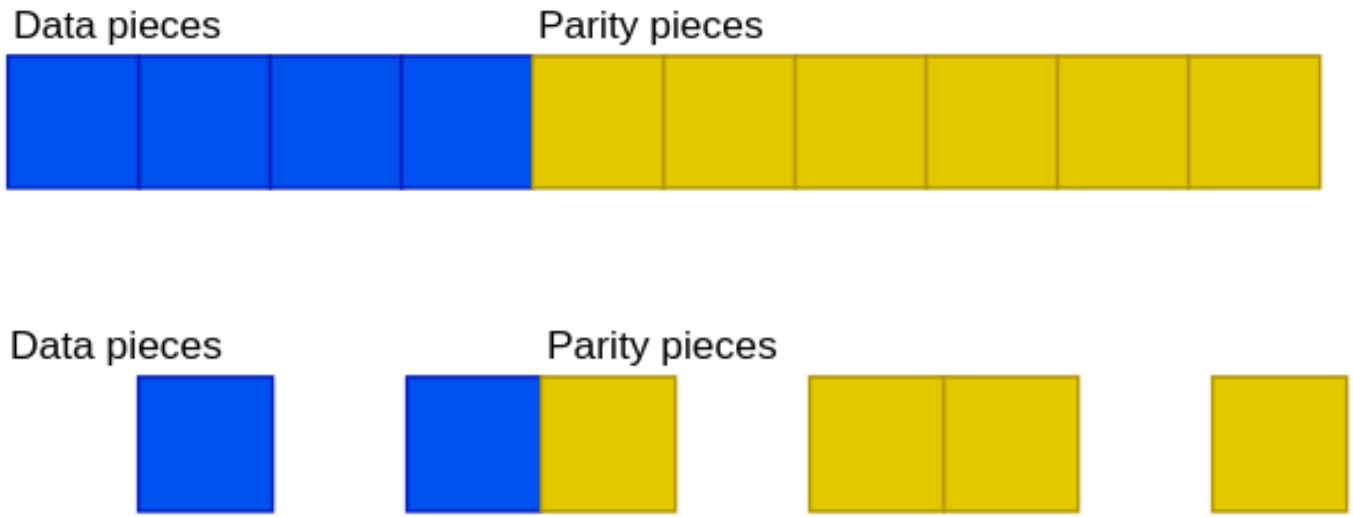
The Storj network enables applications to store data, ensures data reliability, manages access controls, and pays storage nodes.

How It Works

What Happens to Objects?



Erasure coding recap



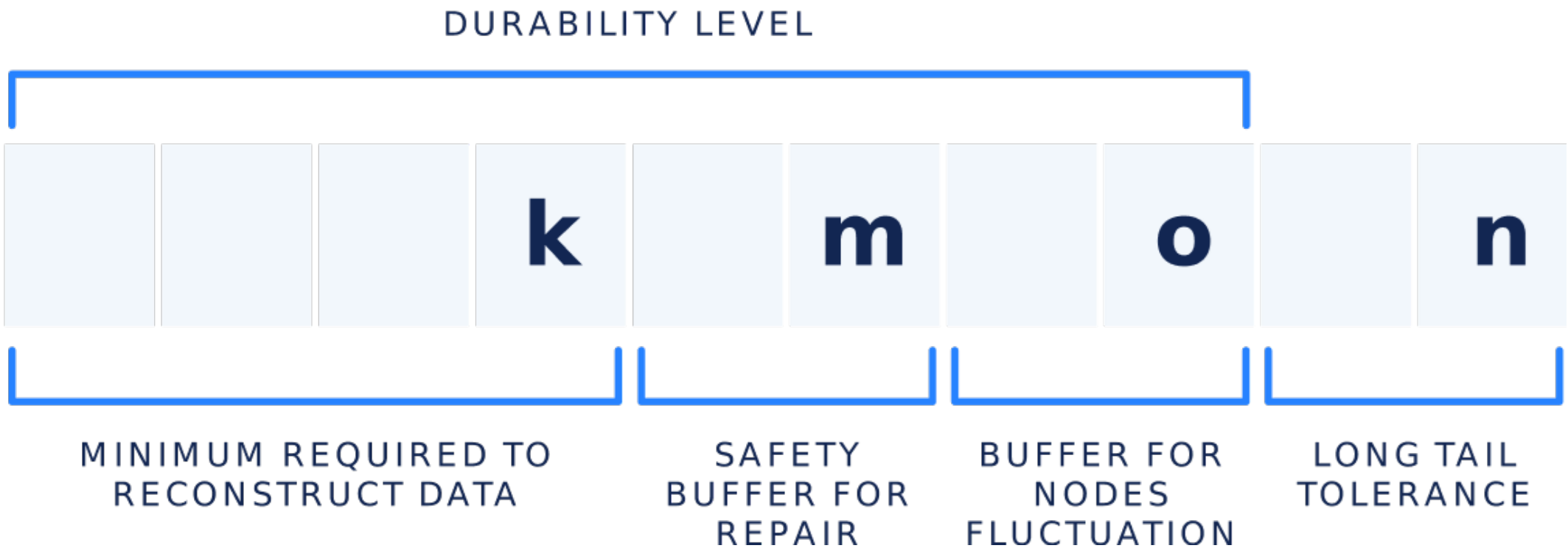

```go
message := "hey hooo this is a string to check EC    "

fec, _ := infectious.NewFEC(4, 10)
err = fec.Encode([]byte(message), func(share infectious.Share) {
        fmt.Println(share.Number)
        fmt.Println(string(share.Data))
    })
```

Output:

```
0
hey hooo t
1
his is a s
2
tring to c
3
heck EC
4
M^5
5
a
6
(
P
7
xDJ
8
!l C
9

]b < C
```

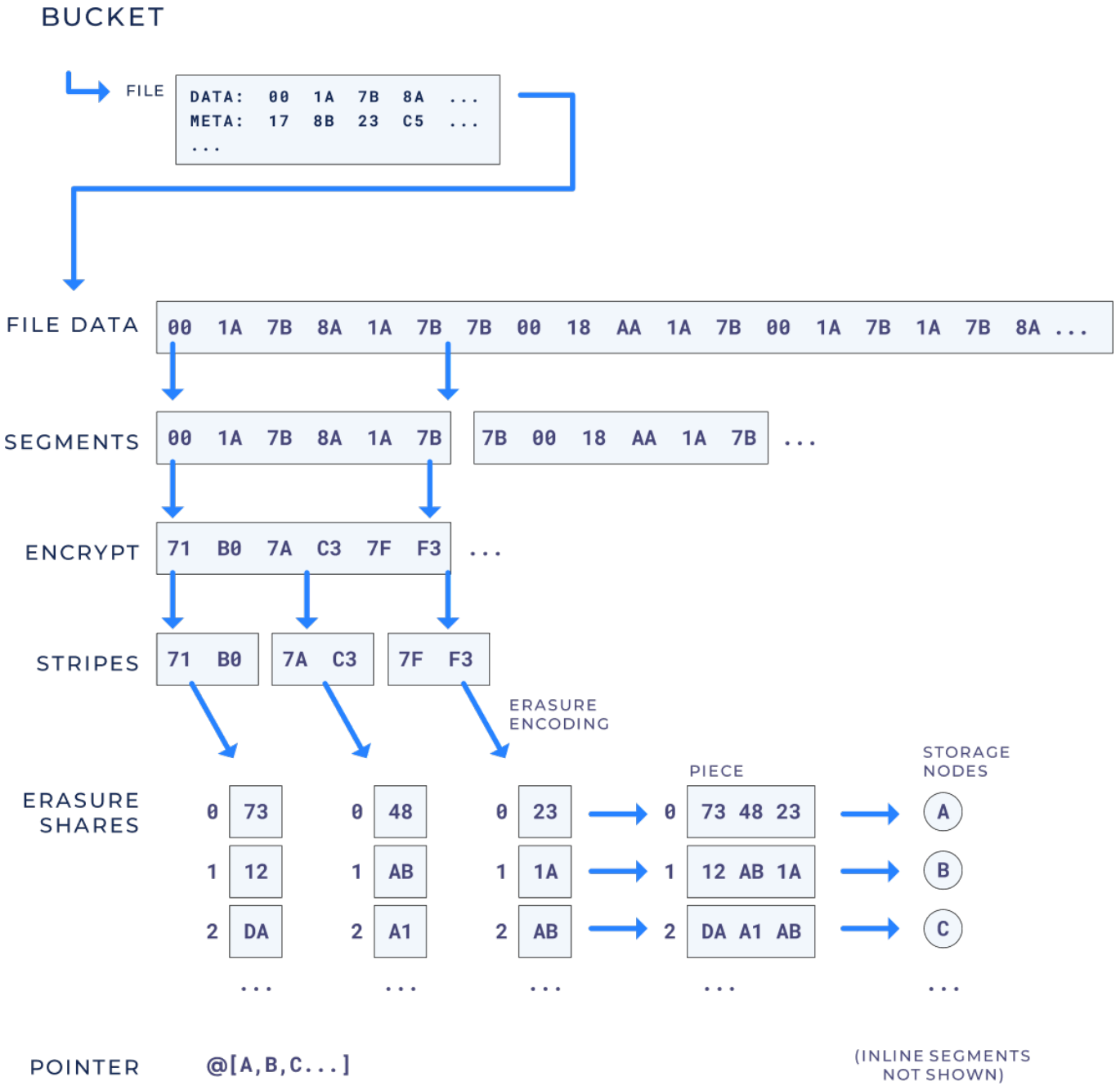
More durability without increased expansion factor

k	n	Exp. factor	$P(D \mid p = 10\%)$
2	4	2	99.2073668132746 16%
4	8	2	99.858686985411326%
8	16	2	99.995462406878260%
16	32	2	99.999994620652776%
20	40	2	99.999999807694154%
32	64	2	99.999999999990544%

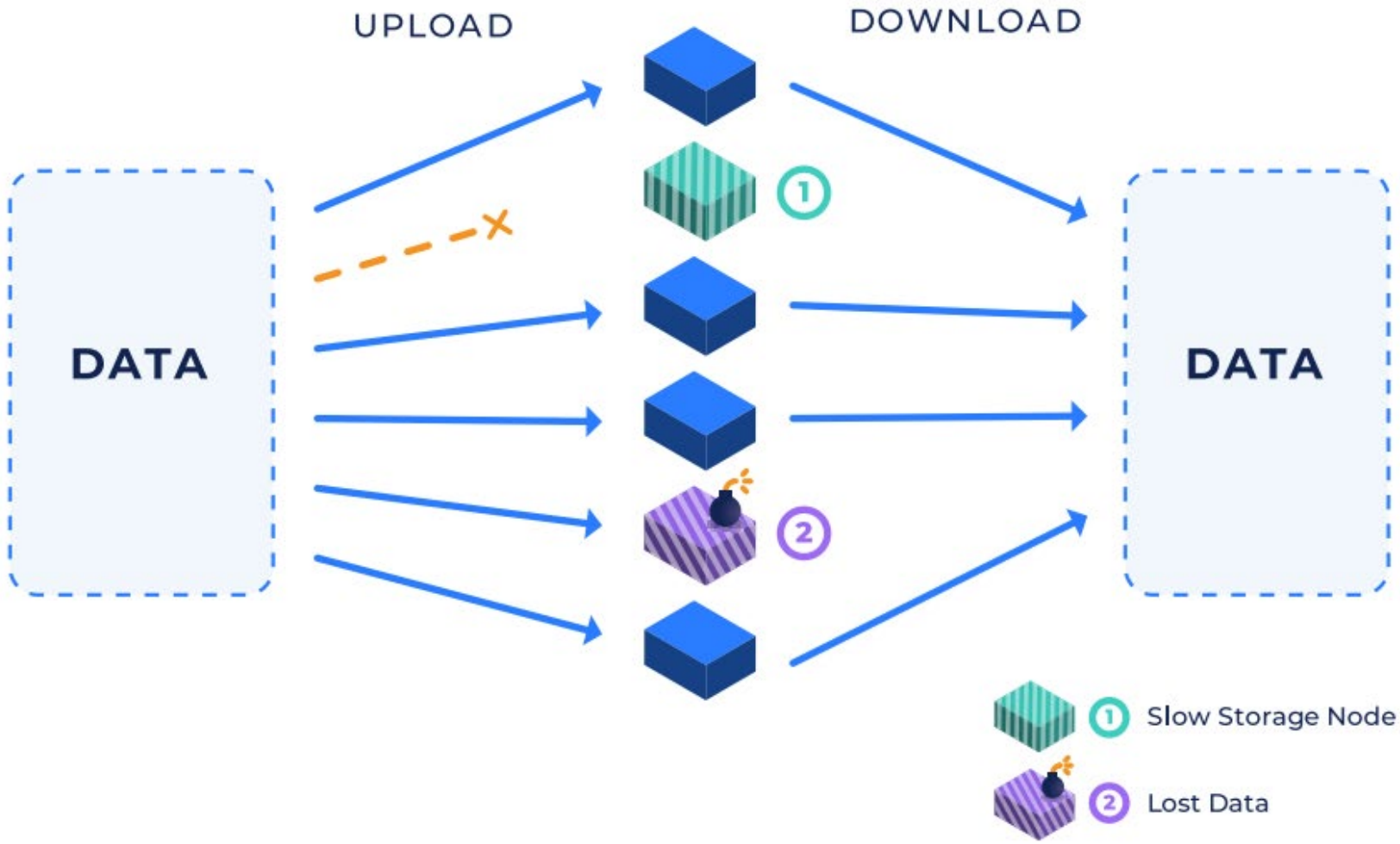
k = minimum required
n = total number of pieces

p = monthly node churn rate

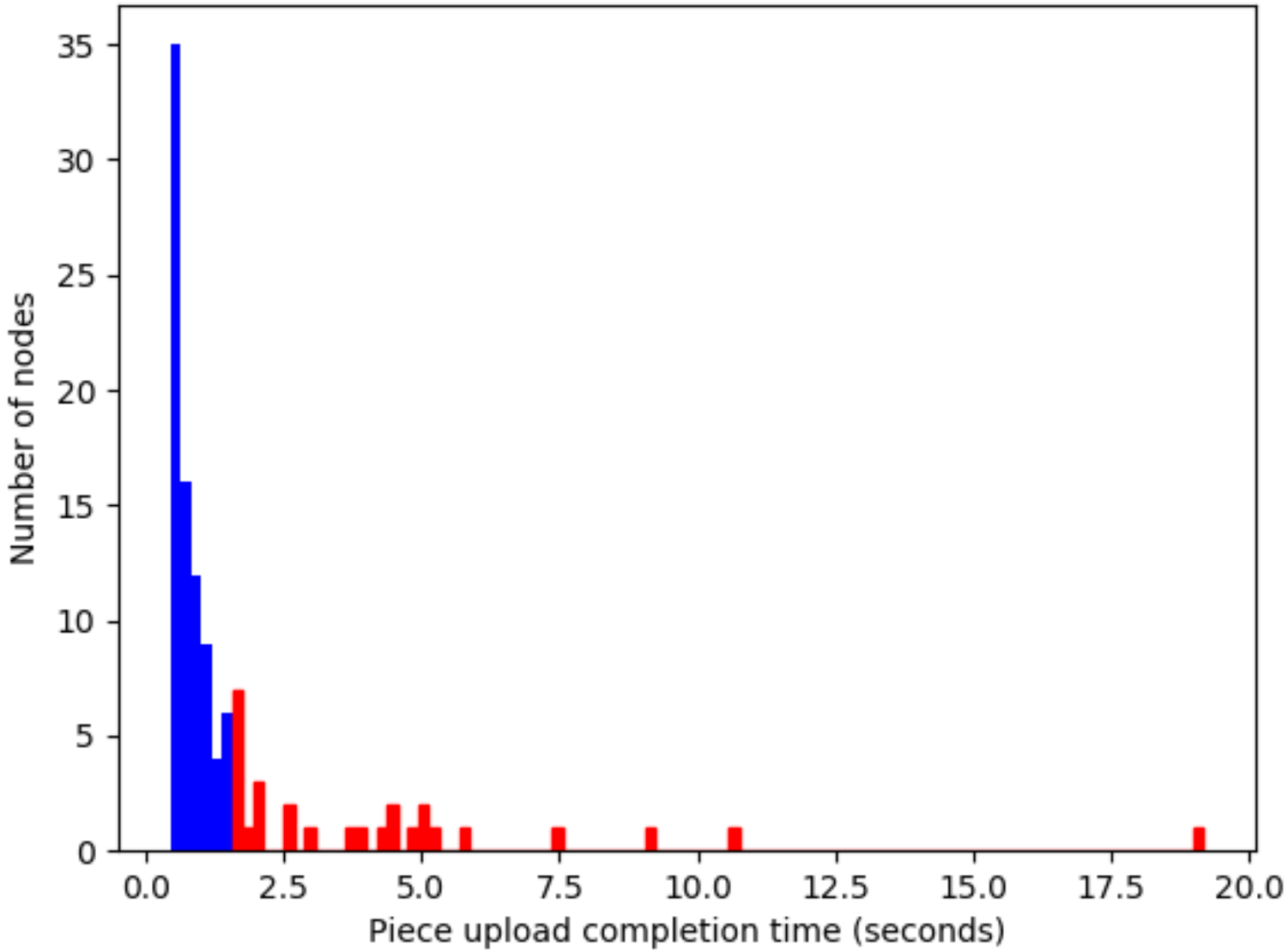
Structure



Data Flow



Long tails



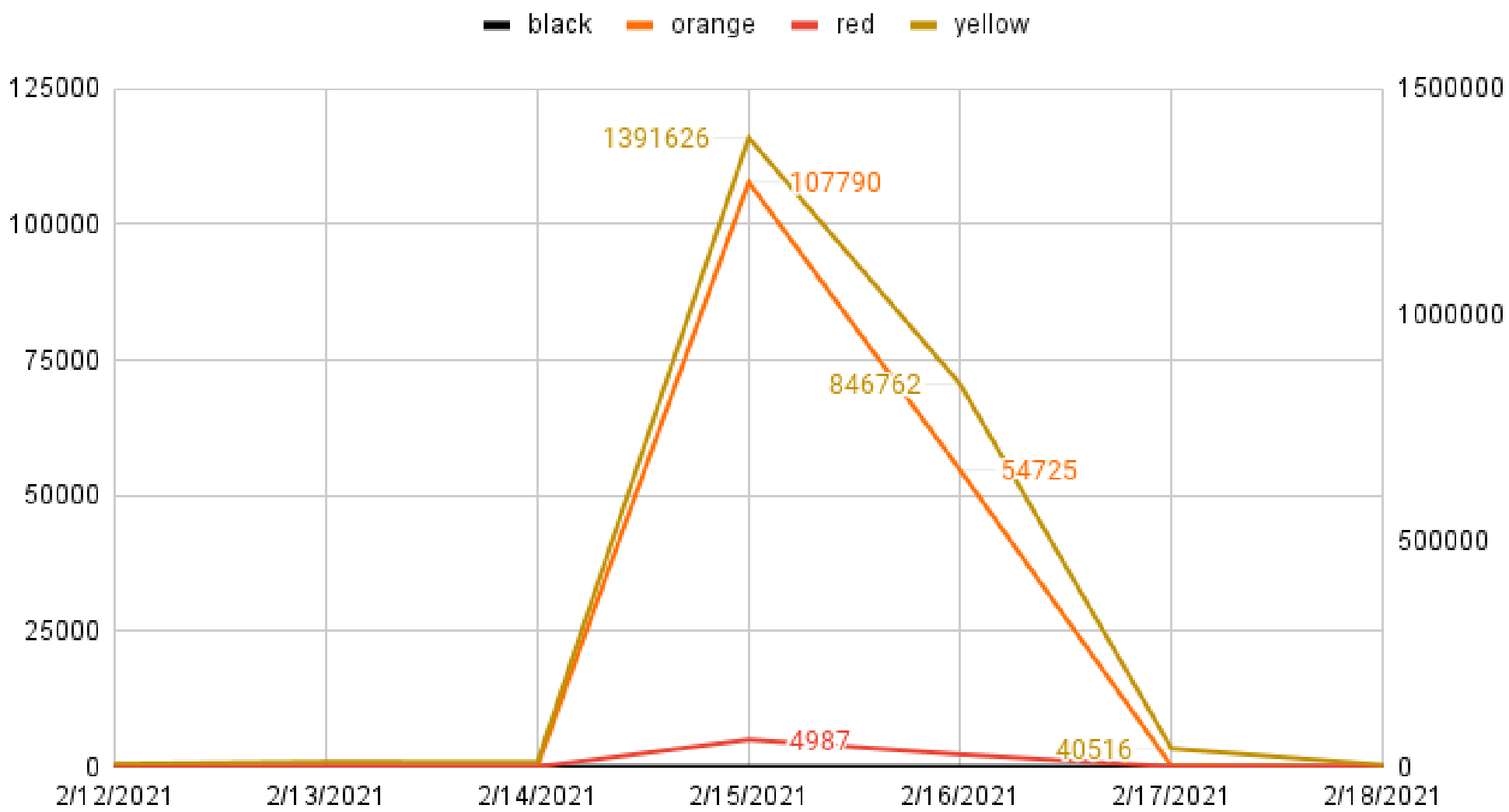
Repair Thresholds

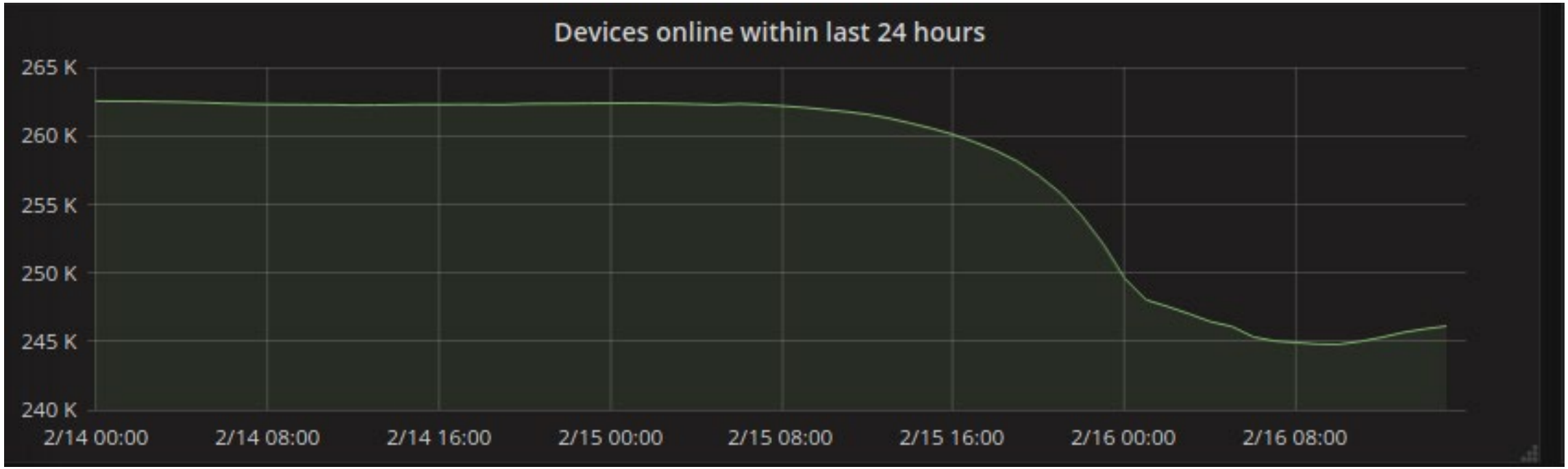
Color	Healthy Piece Count
Green	29 - 30
Yellow	27 - 28
Orange	25 - 26
Red	20 - 24
Black	< 20

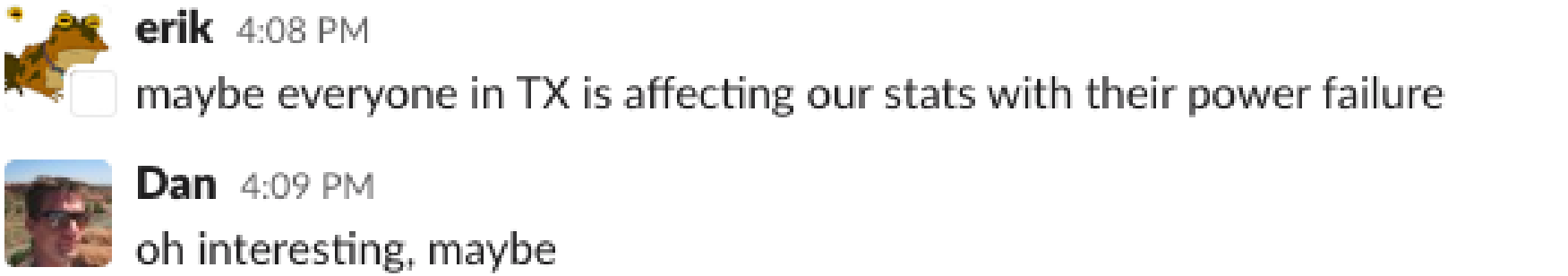
k = 20 (minimum required)
n = 40 (total number of pieces)

Texas Blackout 2021

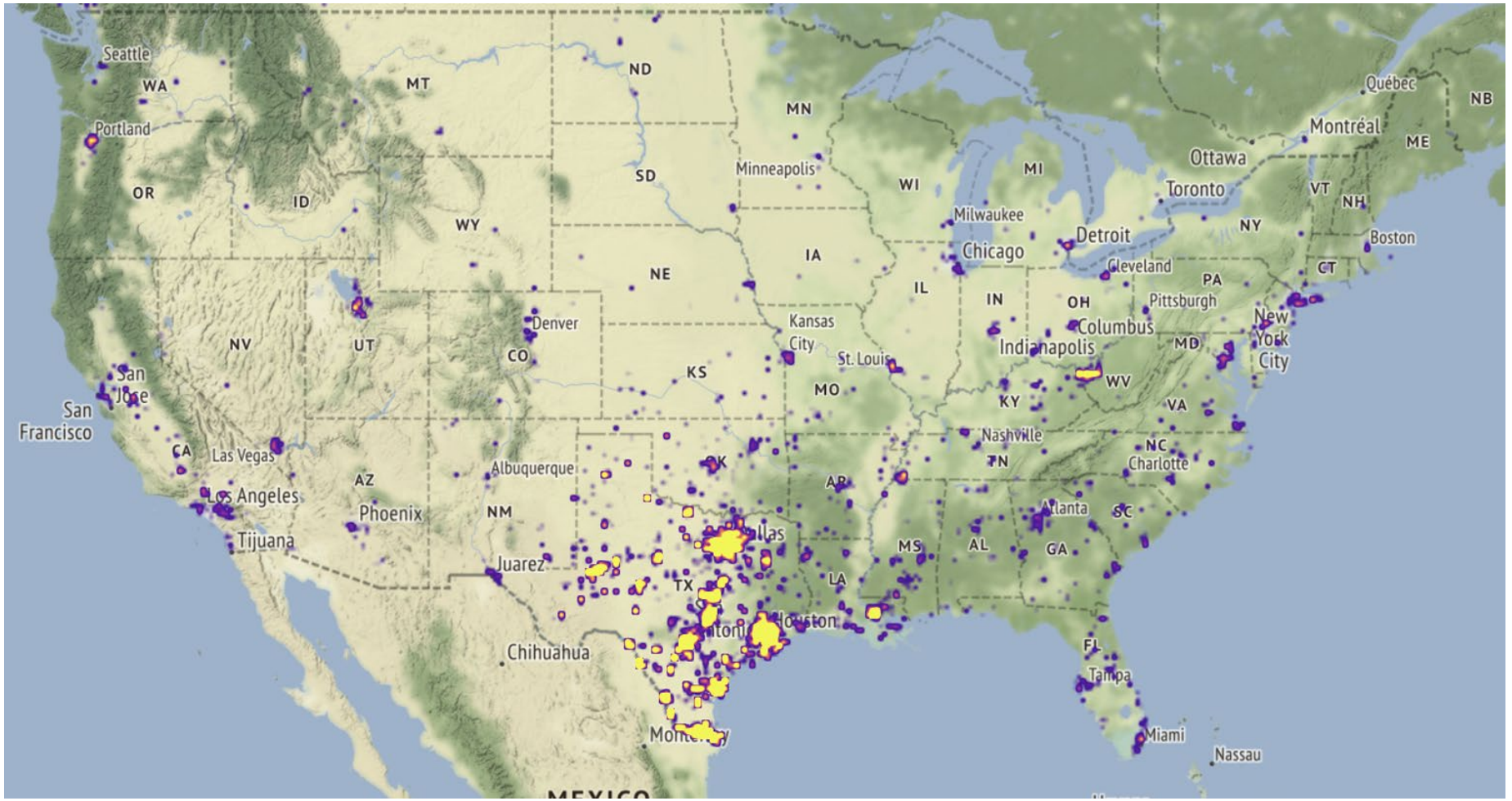
Texas Blackout Repair



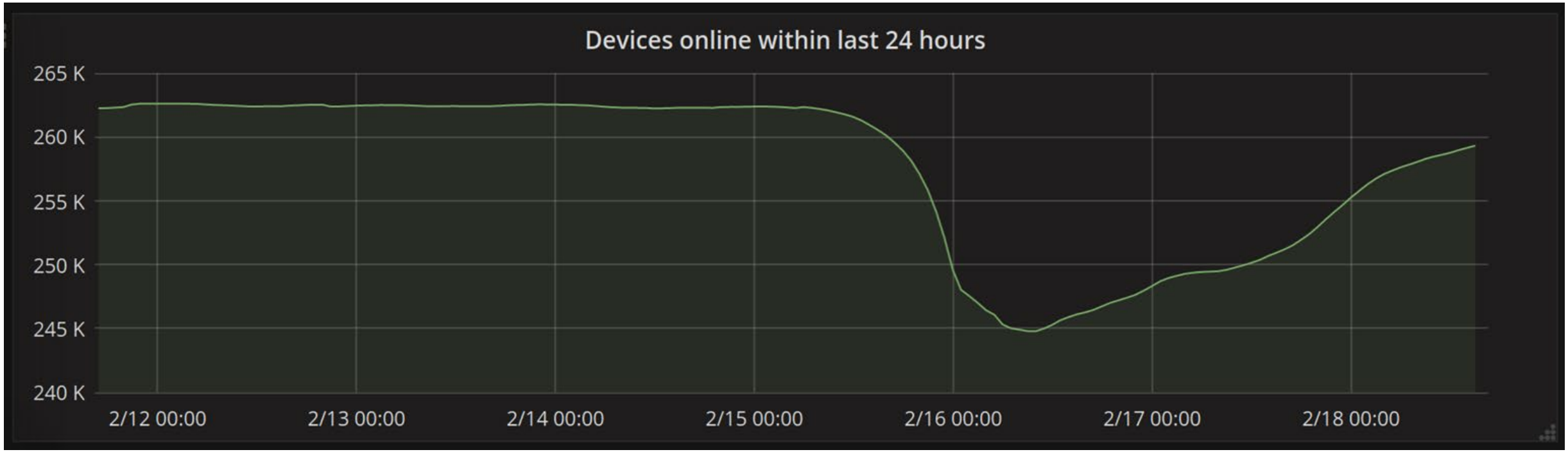




Offline reports



Yellow = Offline



The system works!

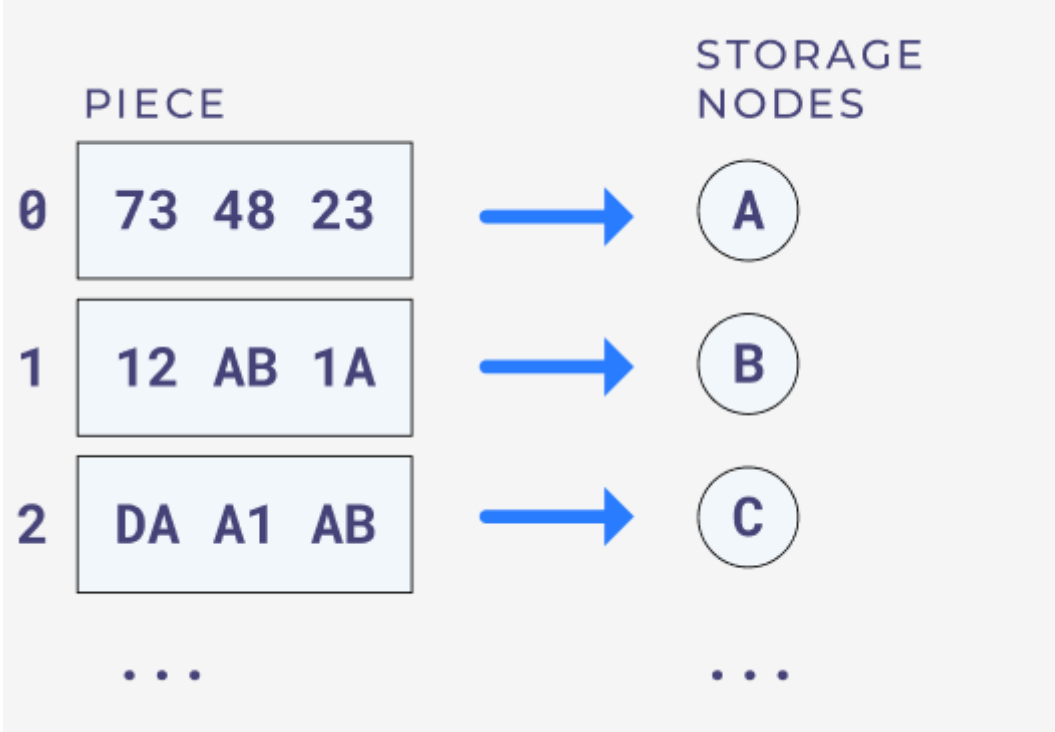
- Durable - 10% of nodes went offline
  - No lost files
- Traffic was diverted automatically
- If it works without Texas, you know it works
  - (Because everything is bigger in Texas)

What makes a lesson learned the hard way?

- Customers were impacted negatively
  - Downtime
  - Corrupted data
- Elusive Bugs that existed for years
  - Hard to replicate
- Race conditions discovered/undiscovered at scale
- Early Architecture/API mistakes balloon to unstable foundations

Duplicate Peer Bug

A repaired piece replaced existing pieces.



Fixing corrupted data

paul 3:32 PM

seems too rare to reproduce easily, we have found 7 bad chunks out of 50million+? (

Can’t reproduce a bug? Add in some Bash

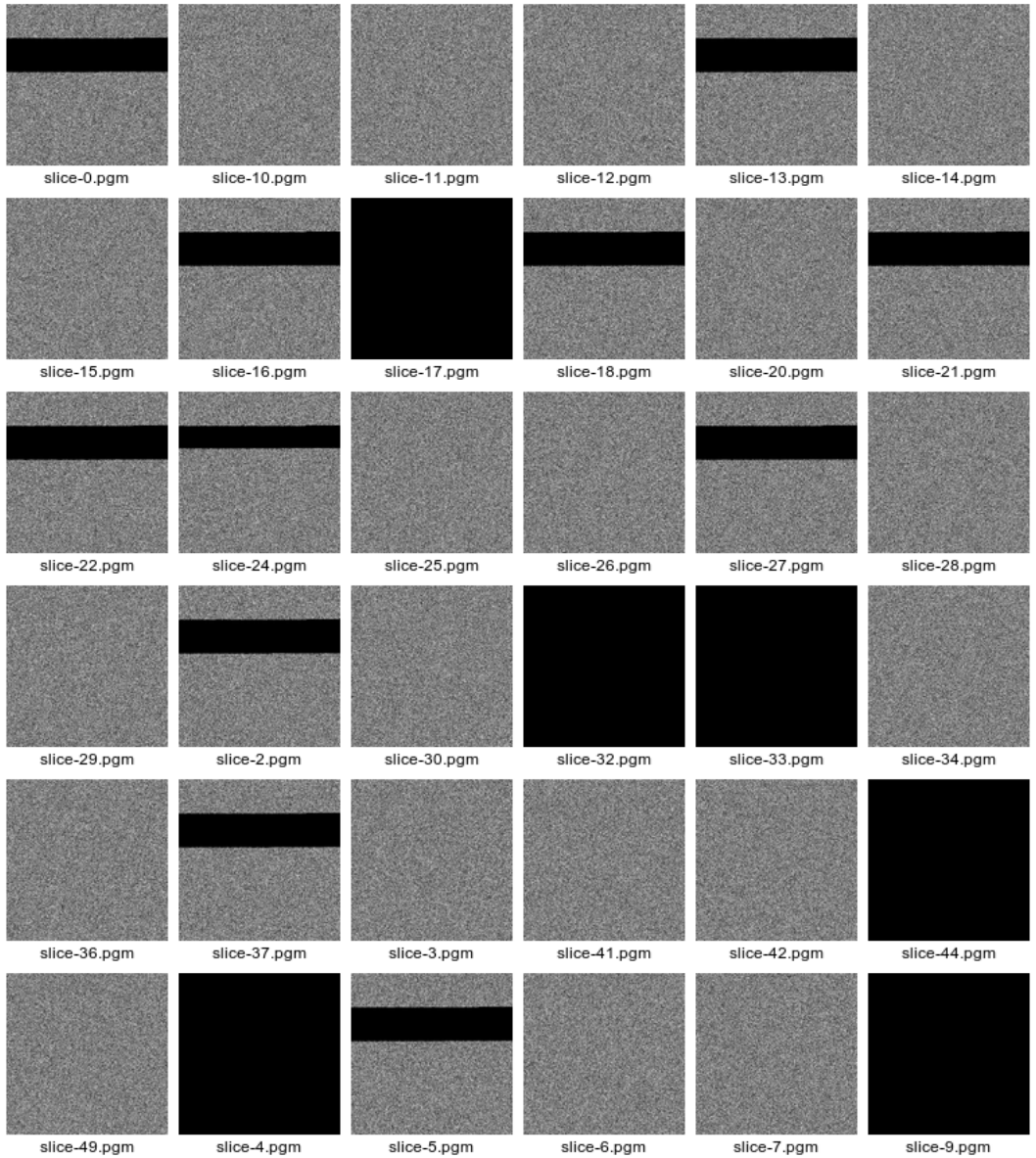
```
# detect_new_chunks
#!/bin/bash
directories=$(ls -d -- /data/disk"$1"/cnl-*/remote/cnl/*/)
/usr/bin/inotifywait -m -e create $directories |
    while read dir action file; do
        #echo "The file '$file' appeared in directory '$dir' via
'$action'"
    echo "$(date +%s) $dir$file" >> new_chunks
    done
```

```
# detect_bad_chunks
#!/bin/bash

while read -r time file; do
  while [ $(( $time - $(date -d "-$1 minutes" +%s) )) -gt 0 ]; do
sleep 1; done
  zeros $file
done <$2
```

```
#fixed peer watches
./detect_new_chunks 42 &
./detect_new_chunks 43 &
./detect_new_chunks 44 &
./detect_new_chunks 45 &
./detect_new_chunks 46 &

./detect_bad_chunks 15 <(tail -f new_chunks) | stdbuf -oL awk '$1 > 32' | tee -a bad_chunks.log
wait
```

How it was happening

```
old_length := stat.Size()
desired_length := max(old_length, offset+length)
io.CopyN(qwh, reader, length)
Truncate(desired_length)
```

- put1 245820
- # 2 minute gap
- put1-retry 245820
- put1-retry done (truncate 294984)
- put2 294984
- put2 done (truncate 344148)
- put1 done (truncate 294984)

Demo

Questions?

Resources

- White paper https://www.storj.io/whitepaper
- Erasure coding https://www.storj.io/blog/replication-is-bad-for-decentralized-storage-part-1-erasure-codes-for-fun-and-profit
- github https://github.com/Storj/

Please take a moment to rate this session.

Your feedback is important to us.

