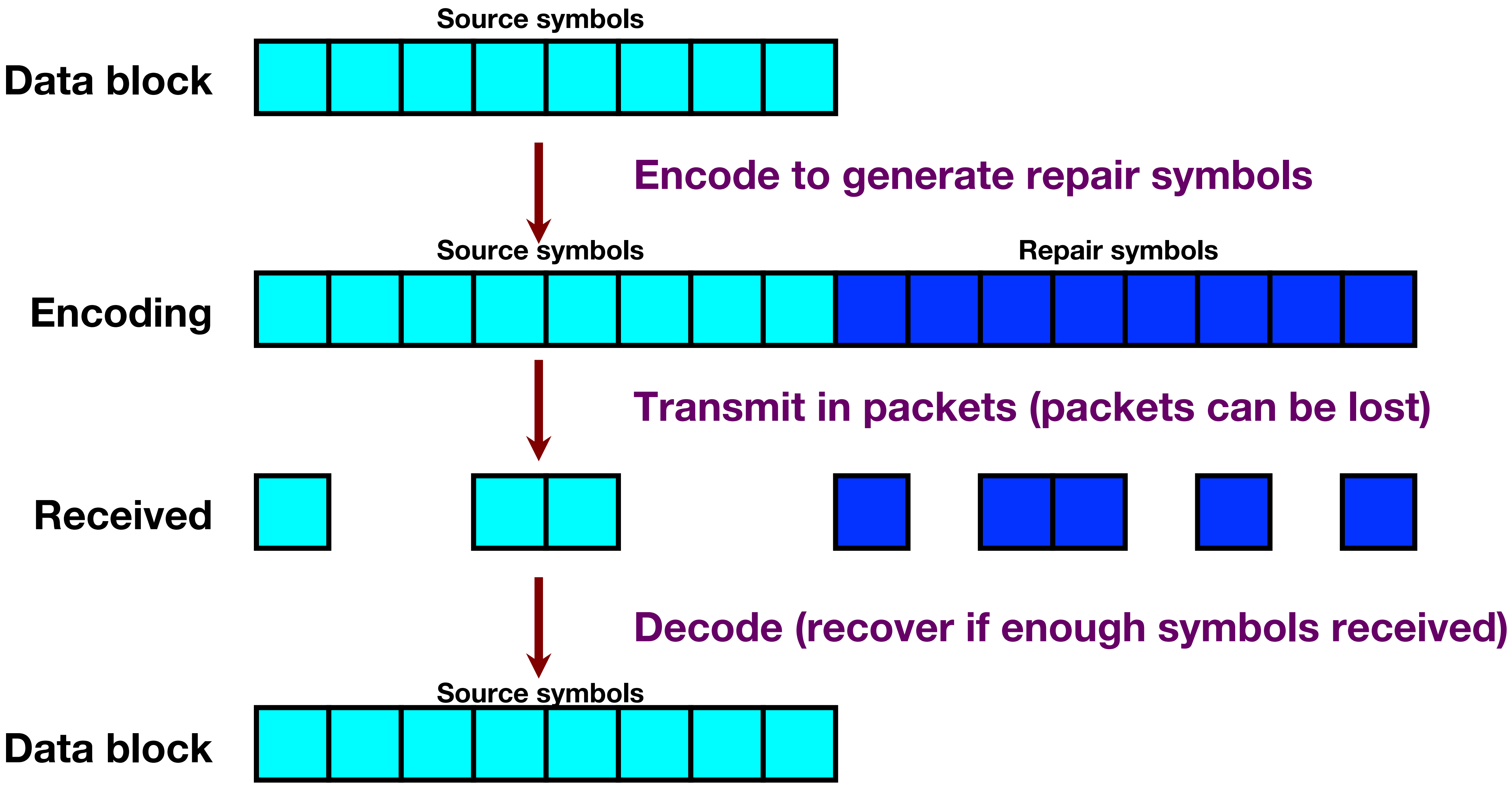
RaptorQ code basics

Michael Luby
International Computer Science Institute

May 2019

Erasure Code



Source symbols

Data block

Encode to generate repair symbols

Source symbols

Repair symbols

Encoding

Transmit in packets (packets can be lost)

Received

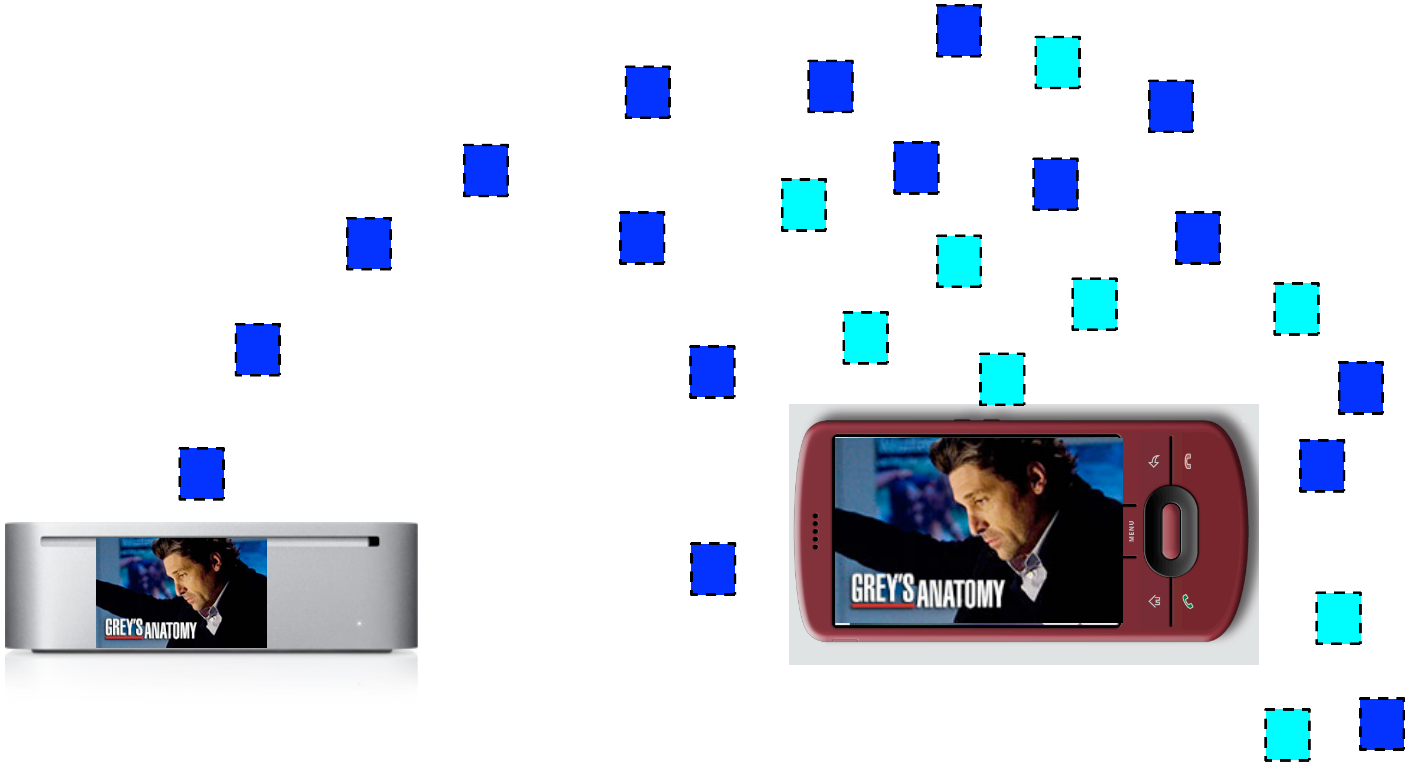
Decode (recover if enough symbols received)

Source symbols

Data block

What is a fountain code?

- ➢ Generate as much encoding as desired, on-the-fly
- ➢ Recover data block from the minimal possible encoding
  - ➢ It doesn’t matter what is received or lost
  - ➢ It only matters that enough encoding is received
  - ➢ Enough is the minimal possible: the size of the data block



RaptorQ code properties

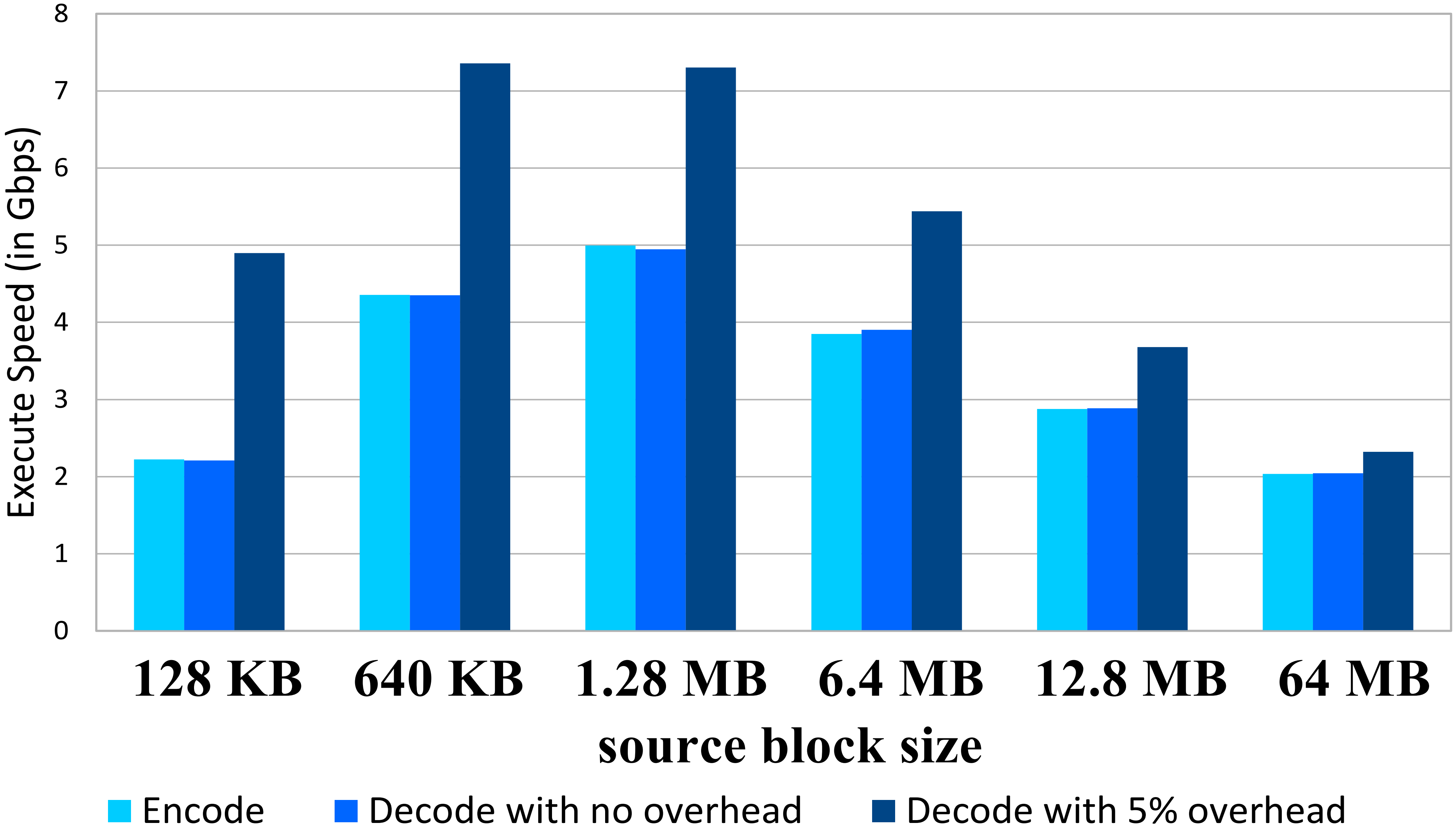
- ➢Fountain code
- ➢Great recovery properties
  - Recovery from any set of symbols in number essentially equal to the number of source symbols in data block
- ➢Linear time encoding and decoding
- ➢Standardized
  - IETF RFC 6330
  - Advanced Television Systems Committee 3.0 (ATSC 3.0)
    - A/331 specifies RaptorQ for packet loss recovery
    - ATSC 3.0 approved for deployment by FCC in November 2019
    - NAB 2019 focus
  - IETF ROUTE protocol (Internet Draft)

Our RaptorQ implementation

- ➢ Fully-compliant with RaptorQ code specified in IETF RFC 6330
- ➢ Supports wide range of parameters
  - Number of source symbols up to 56,403
  - Number of repair symbols up to 2 billion
  - Symbol size up to 64 KB (symbol size typically chosen to fit snugly into a packet)
- ➢ Great recovery properties
  - Recovery from essentially minimal number of symbols
- ➢ Linear time encoding and decoding
  - Initial version will achieve 1+ Gbps on a single core of a PC
  - Later versions will likely achieve 10+ Gbps on a single core of a server-class machine
- ➢ Small image
  - Linux compiled library is ~100 KB
- ➢ Simple and flexible API
  - Same API for encoding, decoding, and transcoding
- ➢ Portable
  - Written in plain C, only needs a C compiler (no other dependencies)
  - Stateless, and thus also reentrant and thread safe
  - No memory allocation
  - No floating point

Speed* of CodornicesRQ

Performance of release 2 on x86-64 platform
(AMD Ryzen 2600 @3.4GHz)
Symbol size = 1280 bytes



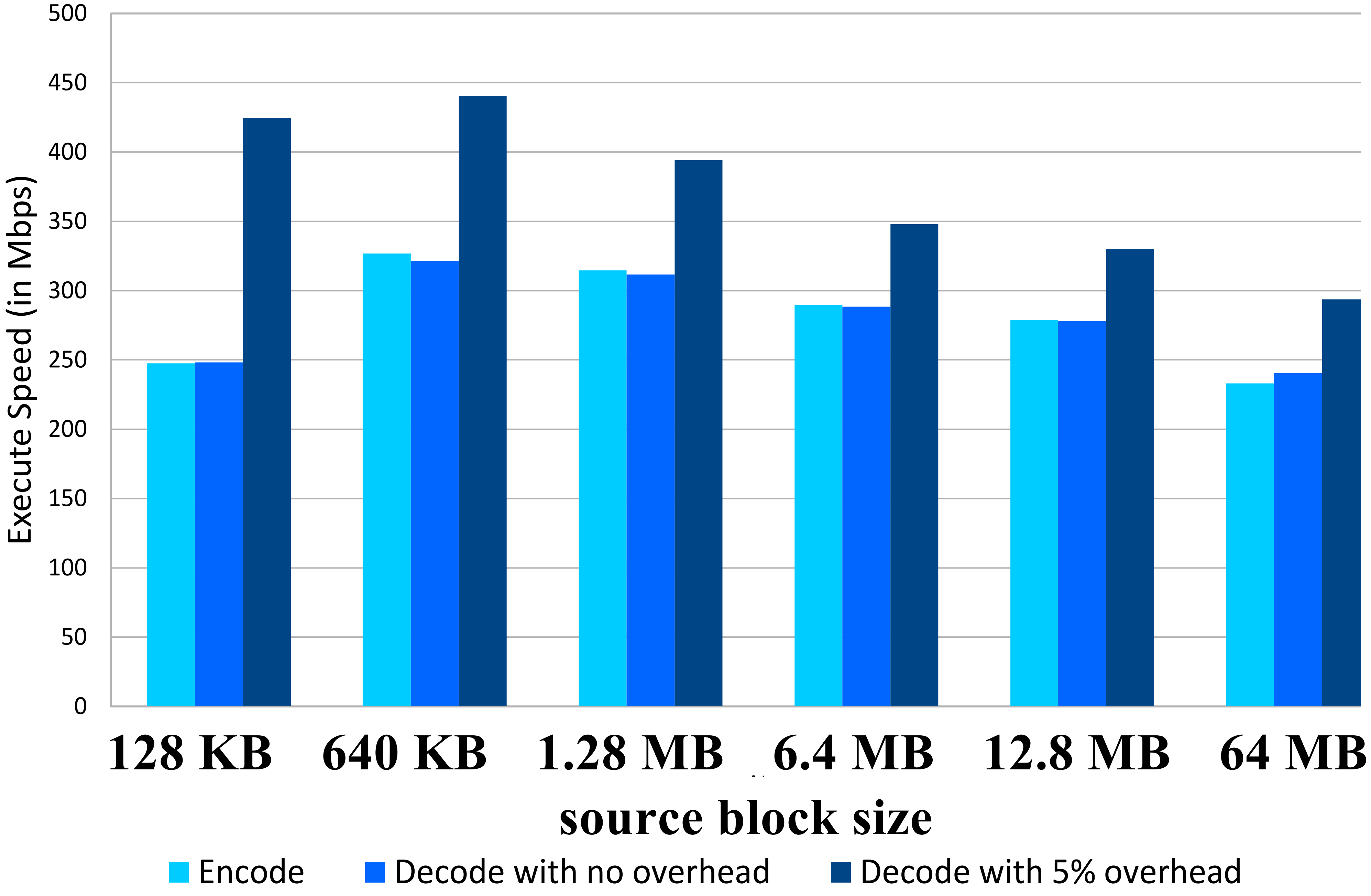
*There will be substantial improvements in future releases

Speed* of CodornicesRQ

Performance of release 2 on ARM platform

(Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz)

Symbol size = 1280 bytes



*There will be substantial improvements in future releases

Some RaptorQ application areas

- ➢End-to-end path protection
  - Protects against intermittent losses along hops of path
  - Reduces latency to reliably deliver data end-to-end
- ➢Broadcast reliable data delivery
  - Reliably deliver data to many receivers over broadcast/multicast channel
  - Reduces the amount of data sent
  - Reduces the amount of time spent to deliver
  - No receiver transmission – enhances receiver LPD
- ➢Multi-path data delivery
  - Reduces latency and more reliable delivery
- ➢Reliable distributed storage
  - Reduces repair bandwidth, reduces storage overhead, more reliable

Example of path protection

Sender

As data arrives at sender for transmission

- Partition data into blocks and source symbols in real-time
- Send source symbols in packets as data arrives without delay
- For example, each block is 1 MB, each symbol is 1 KB (fits into IP packet)

Encode each block in real-time

- Each block consists of 1,000 source symbols (1 MB divided by 1 KB)
- Generate an additional 200 repair symbols using encoder
- Send repair symbols in packets just after the source symbols
- Total of 1,200 symbols sent in packets for each block

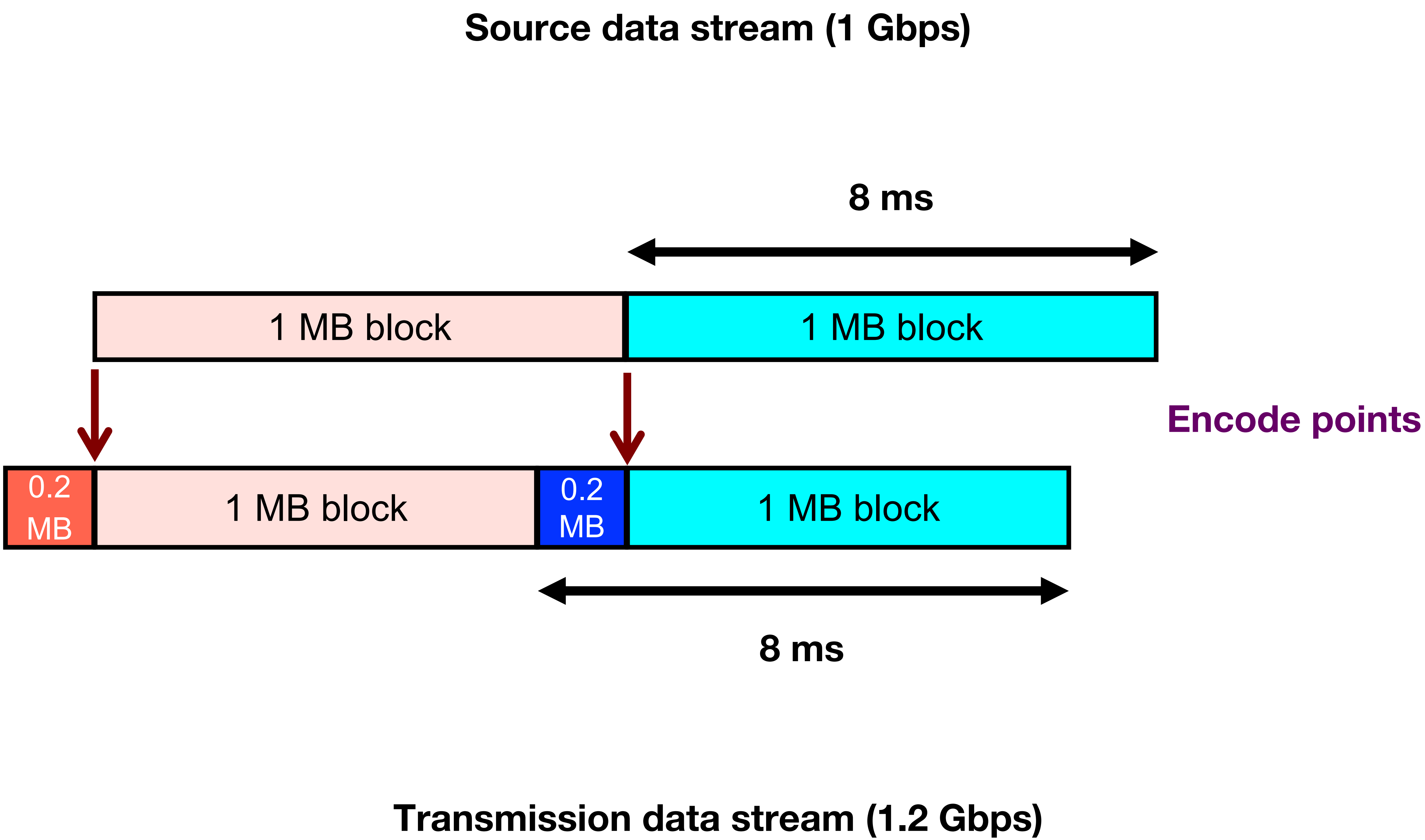
Receiver

Collect received symbols in packets for each block in real-time

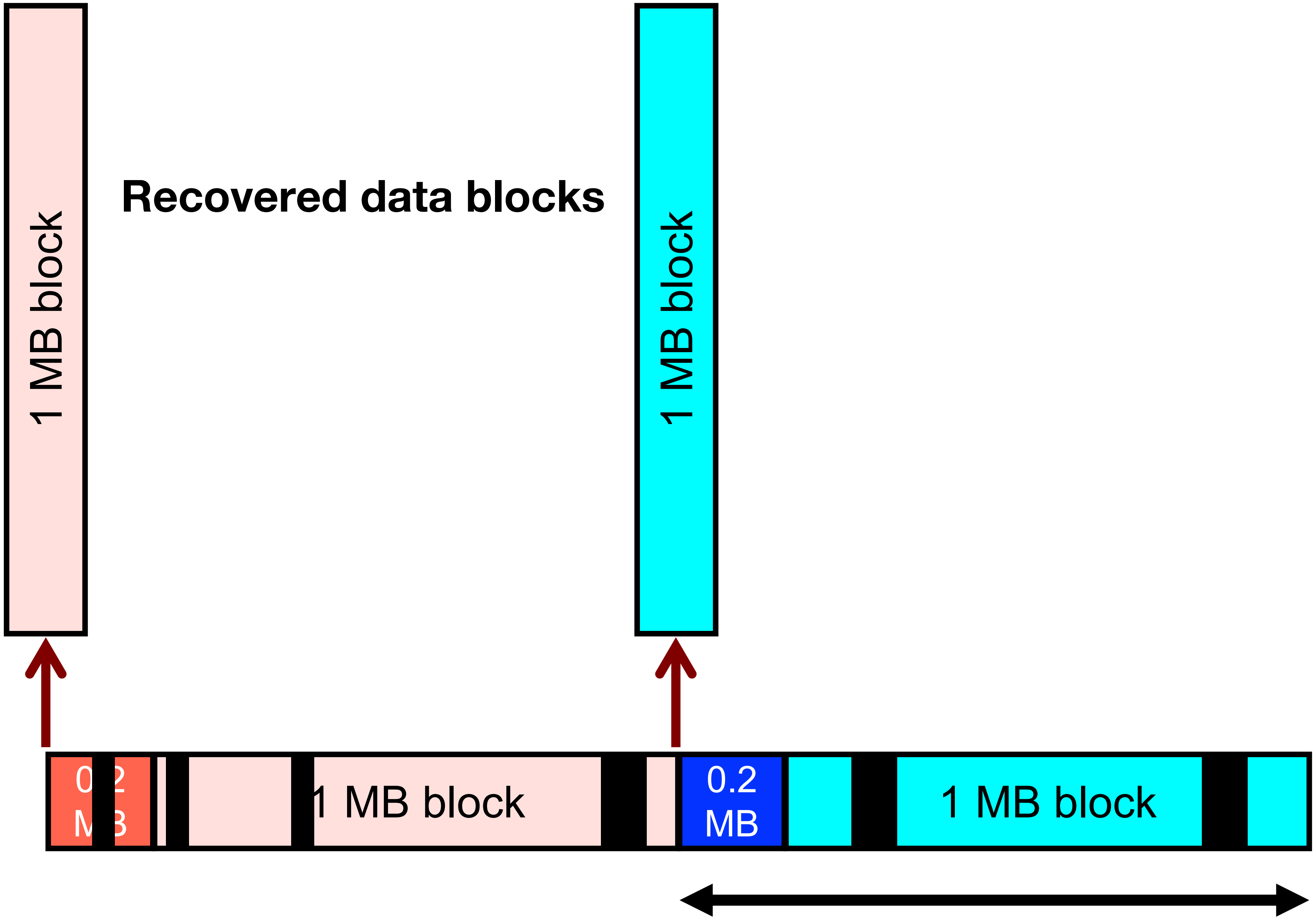
Decode block from received symbols in real-time

- Can recover block if at least 1,000 out of 1,200 symbols are received for block

Path encode/send example



Path decode/receive example



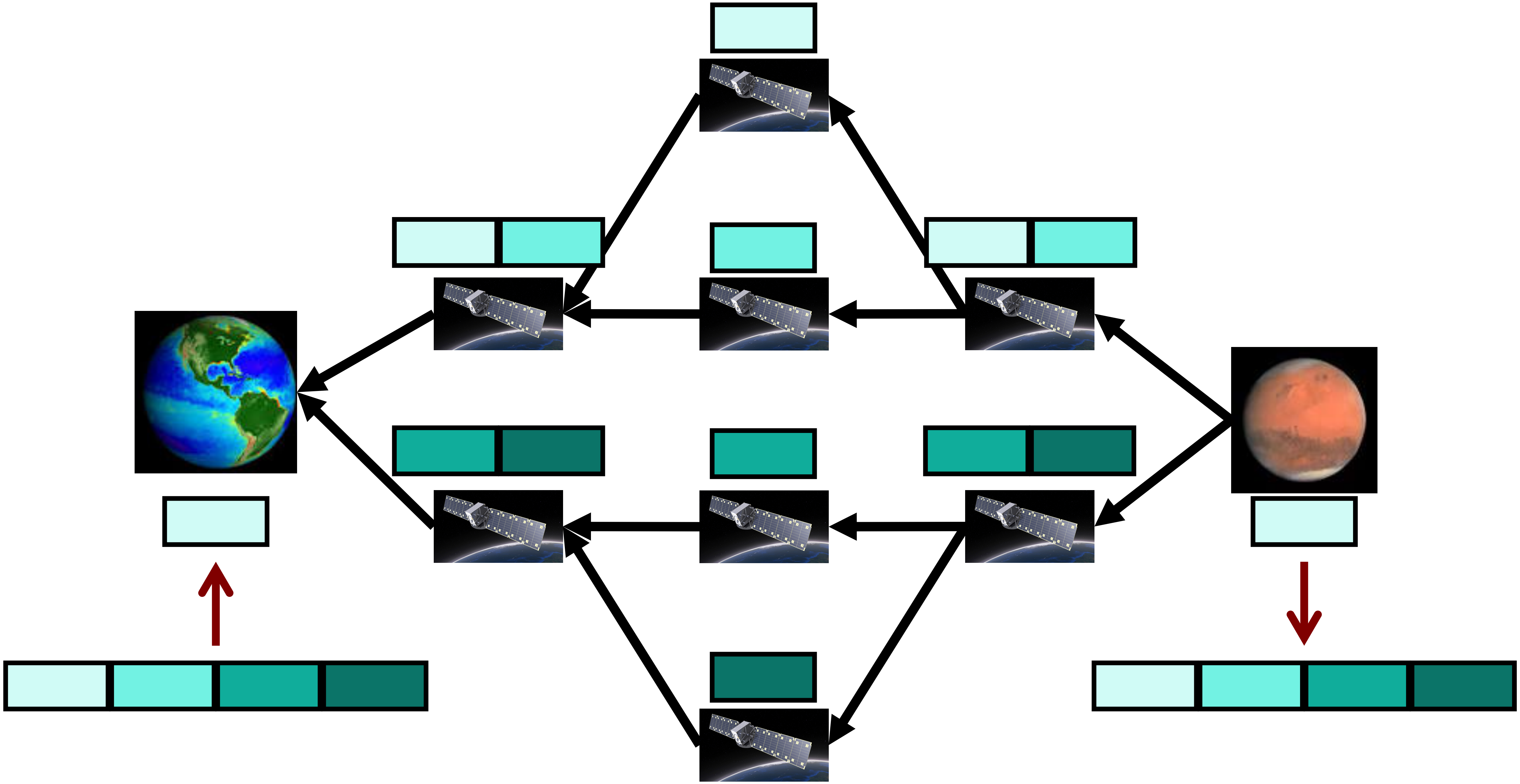
Recovered data blocks

Decode points

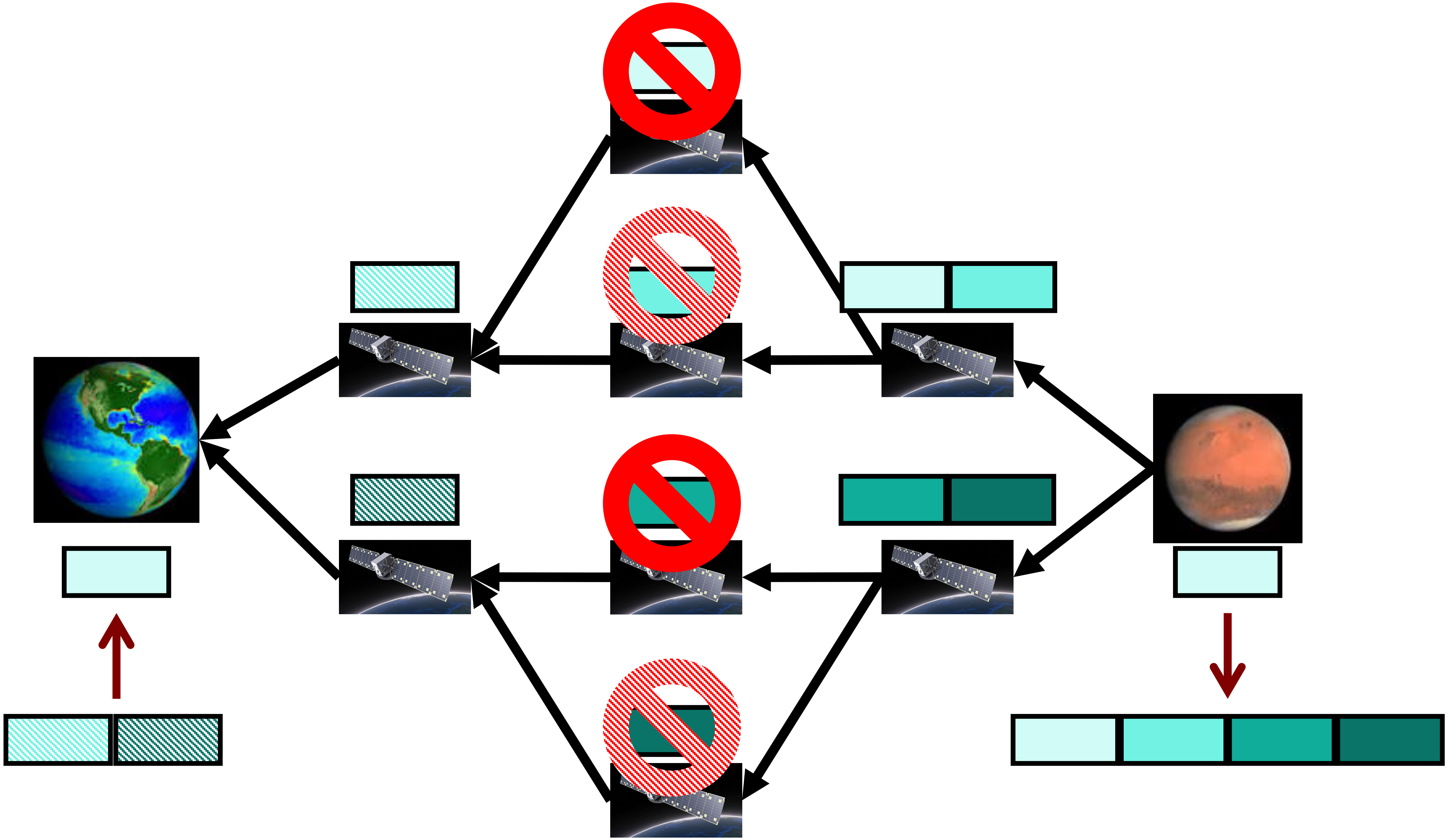
8 ms

Received data stream (1.2 Gbps)

Multipath with store/forward example



Multipath with store/forward example



Thank you!

Go to www.codornices.info
for more information