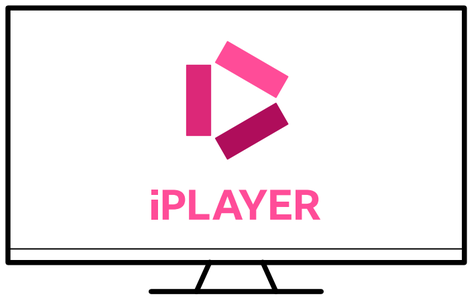
Towards a higher throughput media CDN

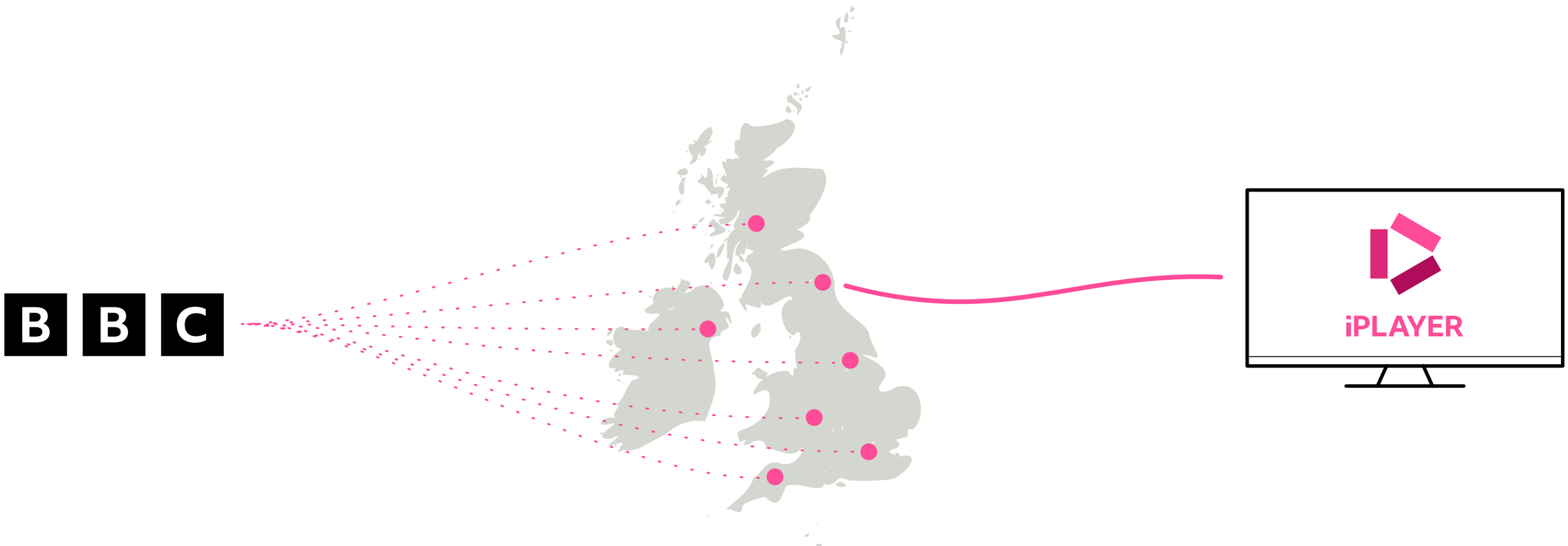
Jonathan Heathcote • BBC Research & Development • NetUK 2024

BBC

The BBC's Media CDN

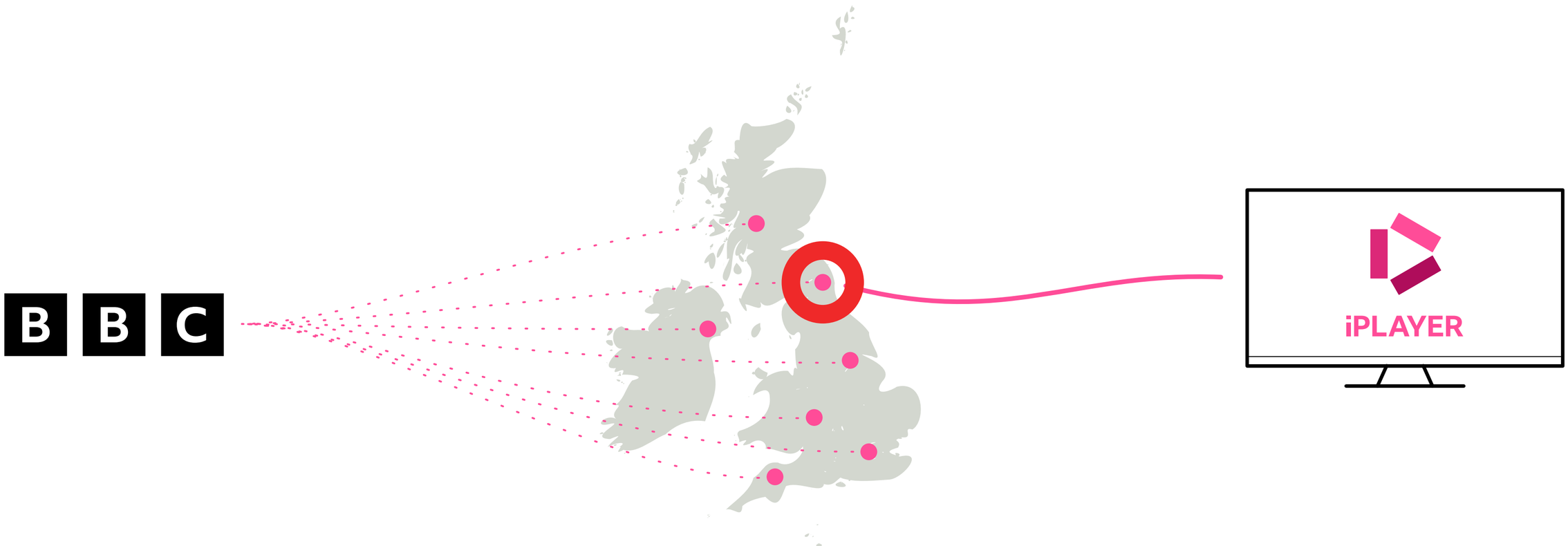


The BBC's Media CDN



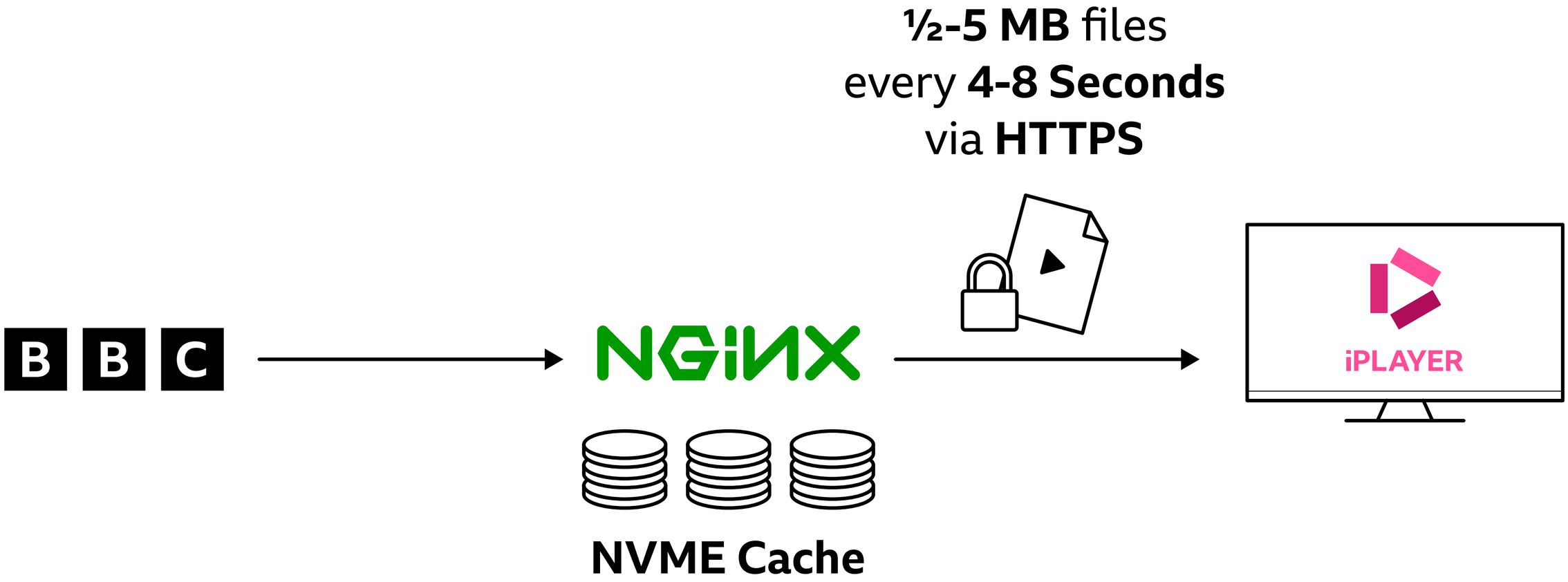
BBC

The BBC's Media CDN

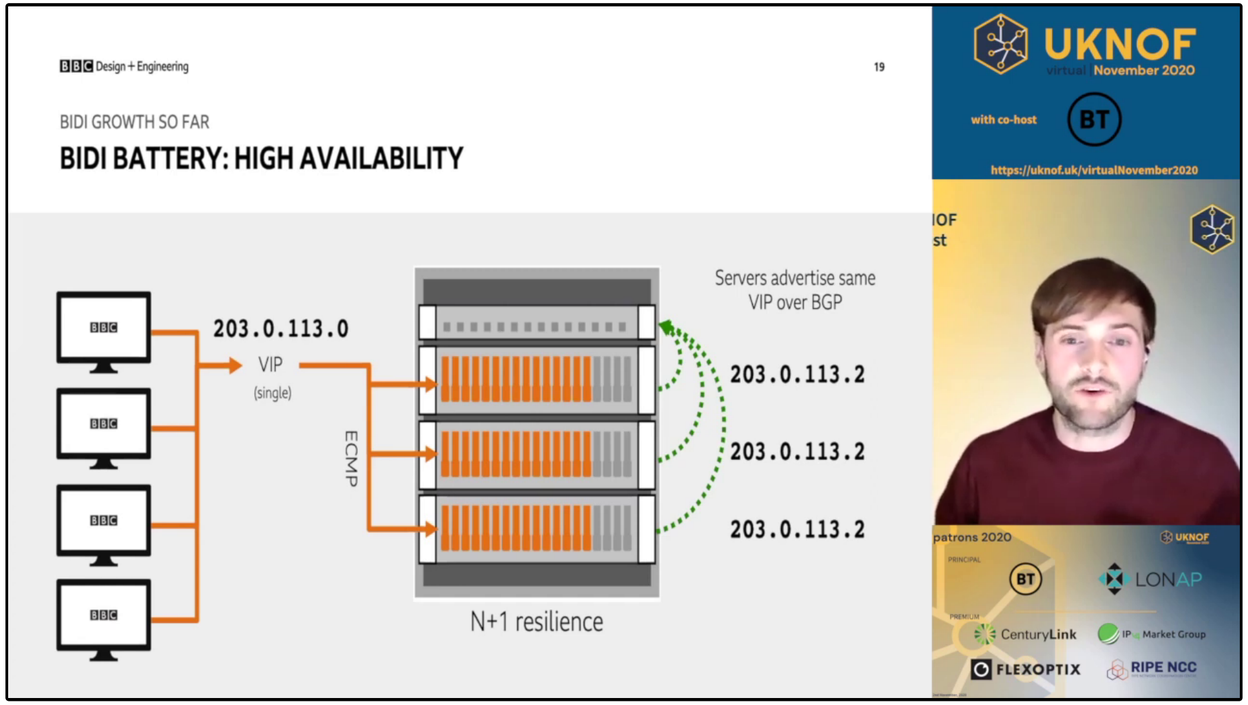


BBC

Edge cache workload

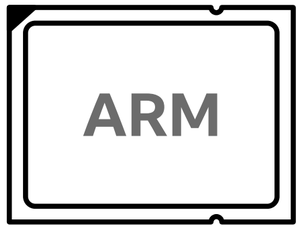


See also...

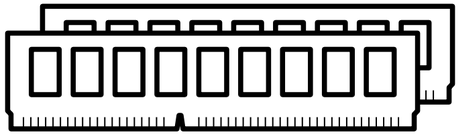


“**Building the BBC's media CDN**” by Alistair Wooldrige from UKNOF 2020

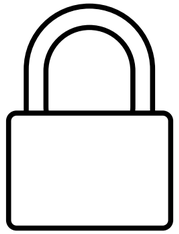
This talk



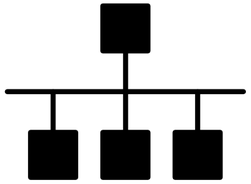
ARM Servers



Memory Bandwidth



kTLS & Offload



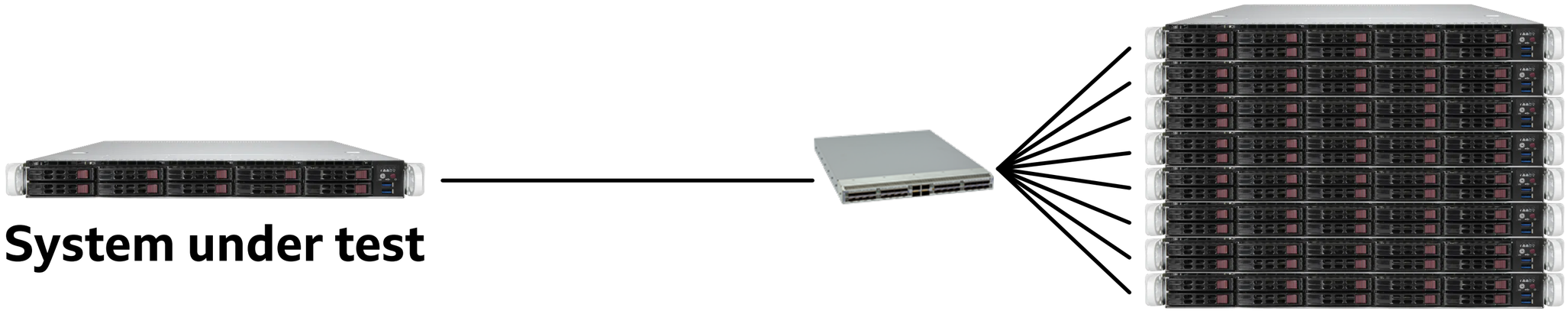
Network Behaviour

BBC

ARM Servers

BBC

Test lab



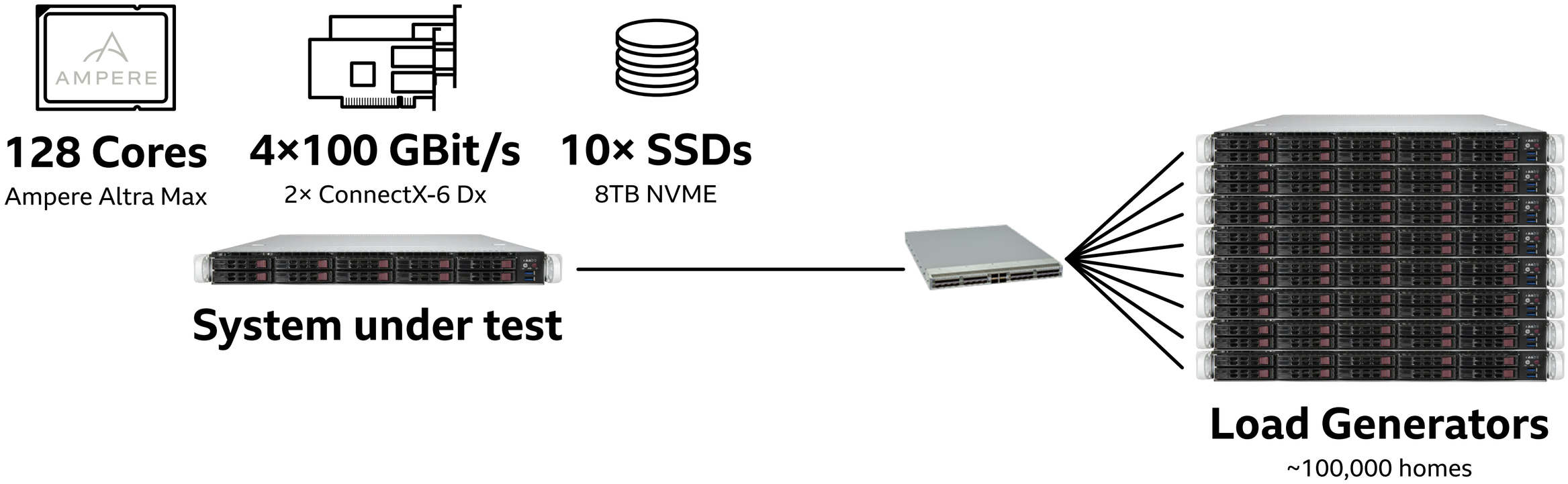
System under test

Load Generators

~100,000 homes

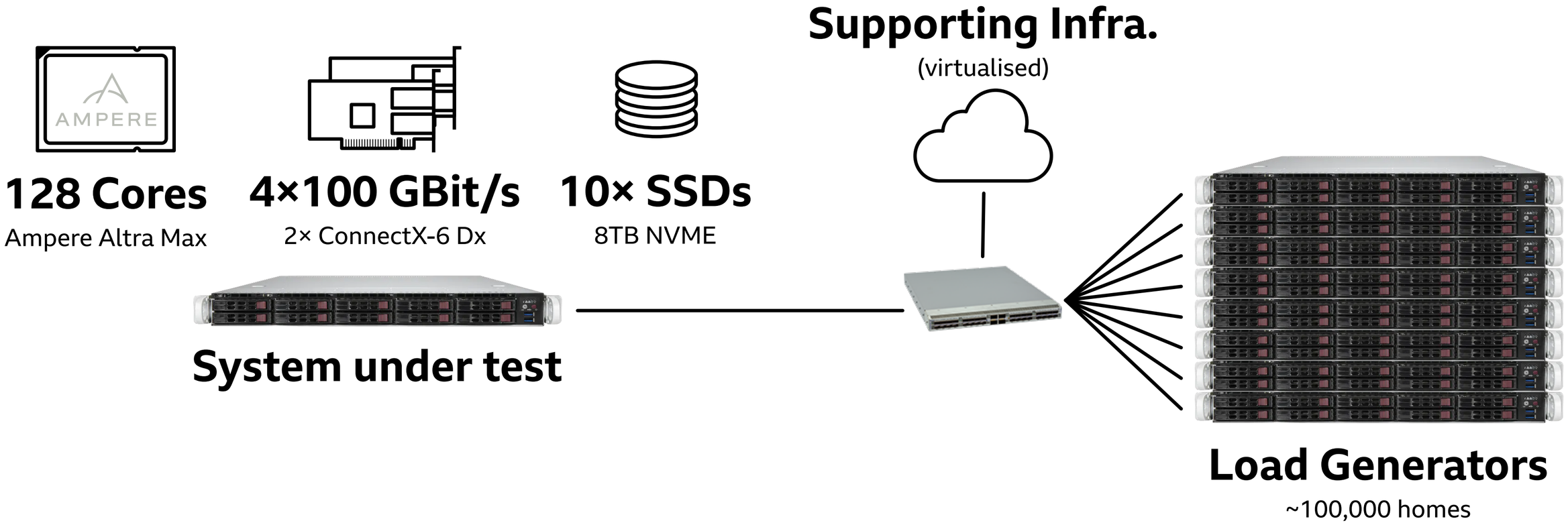
BBC

Test lab


AMPERE
128 Cores
Ampere Altra Max
4×100 GBit/s
2× ConnectX-6 Dx
10× SSDs
8TB NVME
System under test
Load Generators
~100,000 homes

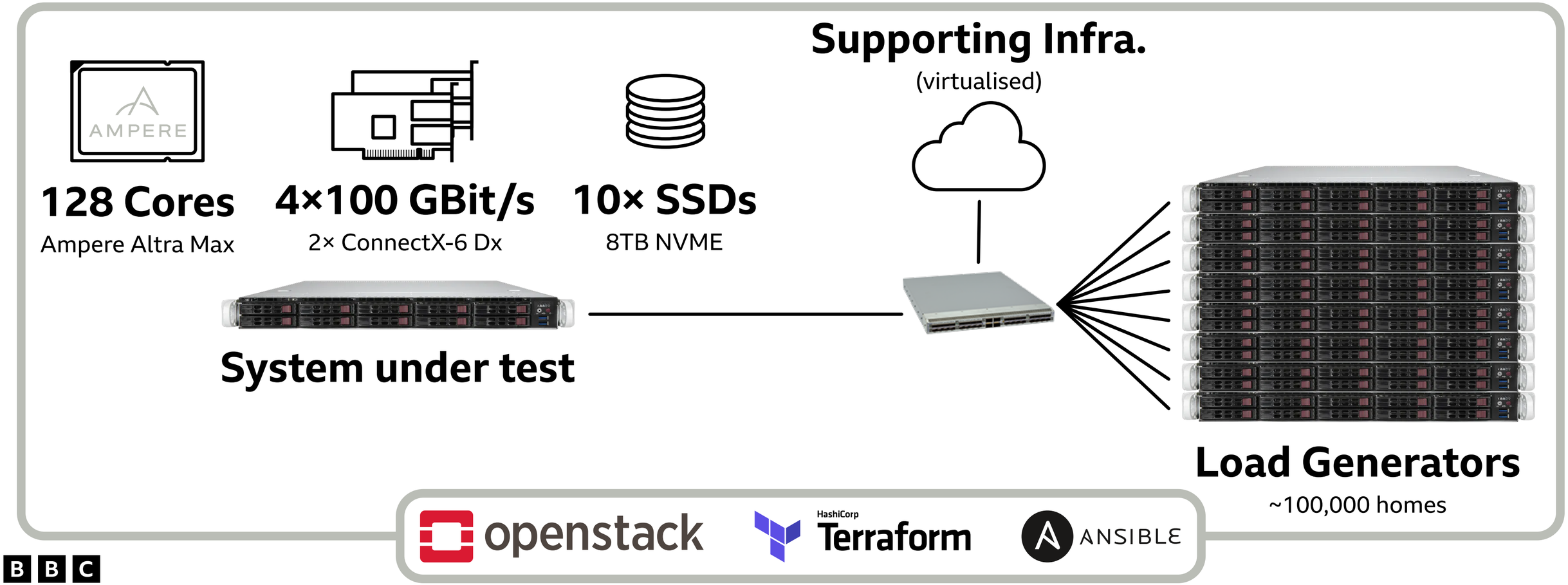
BBC

Test lab


AMPERE
128 Cores
Ampere Altra Max
4×100 GBit/s
2× ConnectX-6 Dx
10× SSDs
8TB NVME
Supporting Infra.
(virtualised)
System under test
Load Generators
~100,000 homes

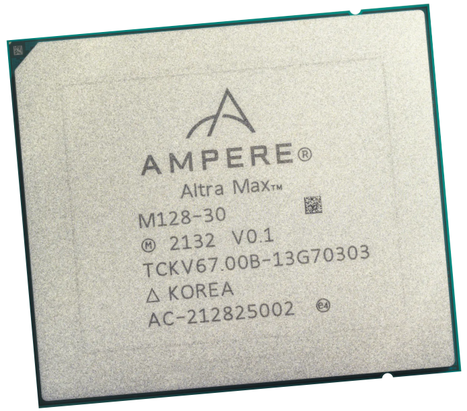
BBC

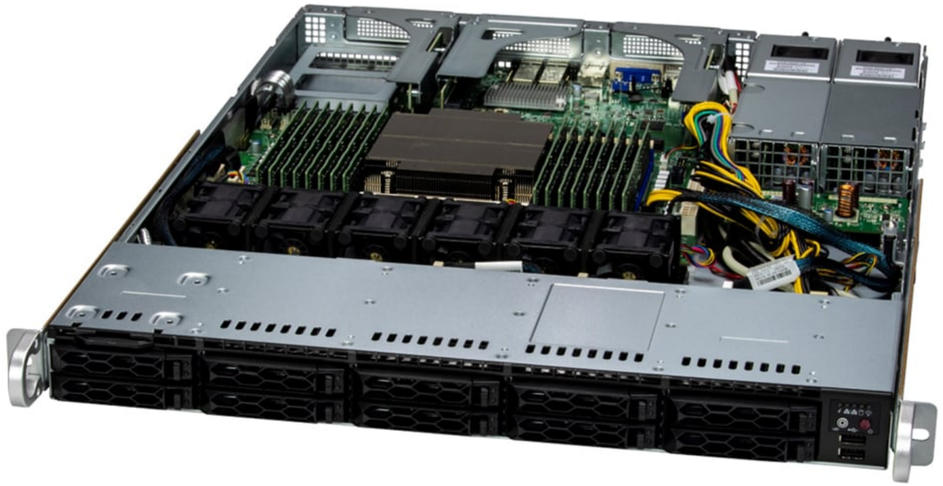
Test lab


AMPERE
128 Cores
Ampere Altra Max
4×100 GBit/s
2× ConnectX-6 Dx
10× SSDs
8TB NVME
Supporting Infra.
(virtualised)
System under test
Load Generators
~100,000 homes
openstack
HashiCorp
Terraform
ANSIBLE

BBC

ARM Servers





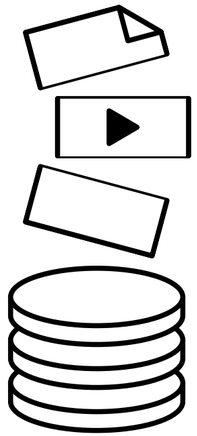
BBC

Operating System Page Cache Primer

NGINX

BBC

Operating System Page Cache Primer



NGINX

BBC

Operating System Page Cache Primer

NGINX

BBC

Operating System Page Cache Primer

BBC

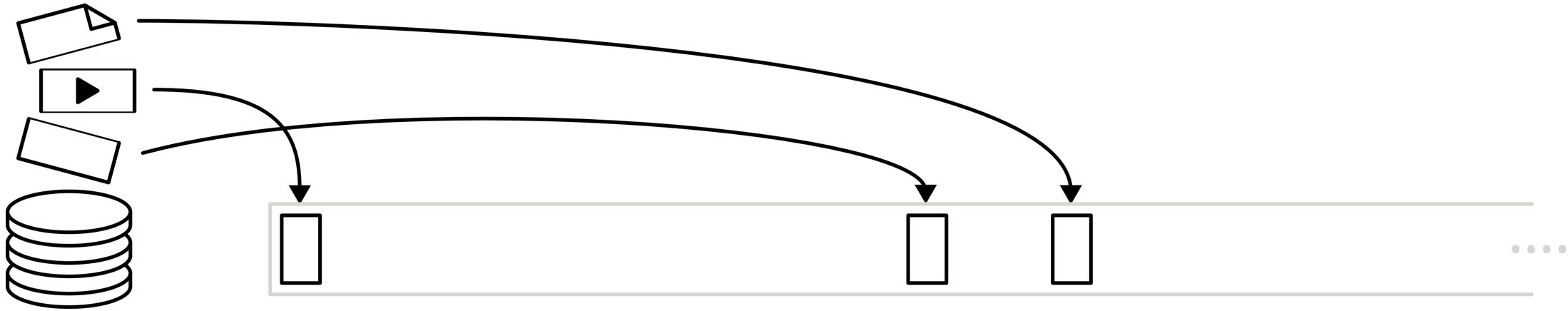
NGINX

Operating System Page Cache Primer

NGINX

BBC

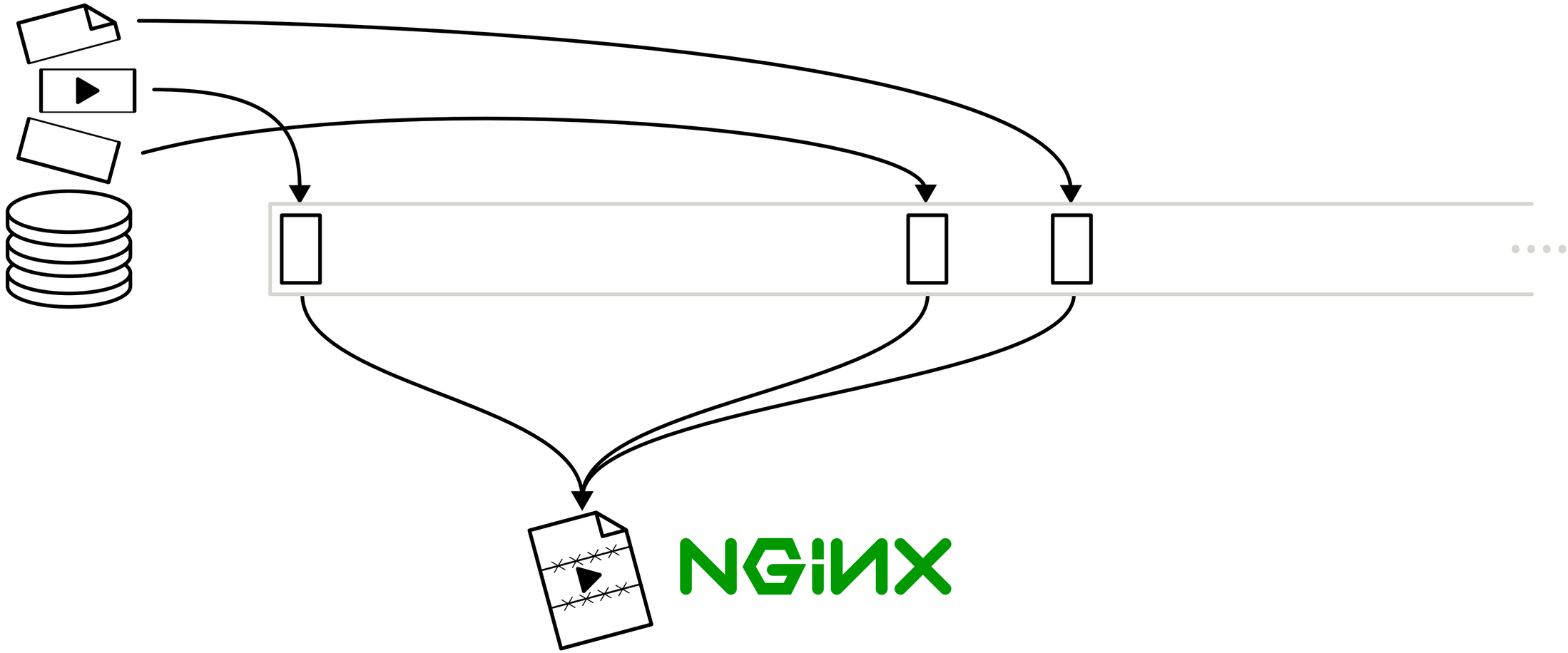
Operating System Page Cache Primer



BBC

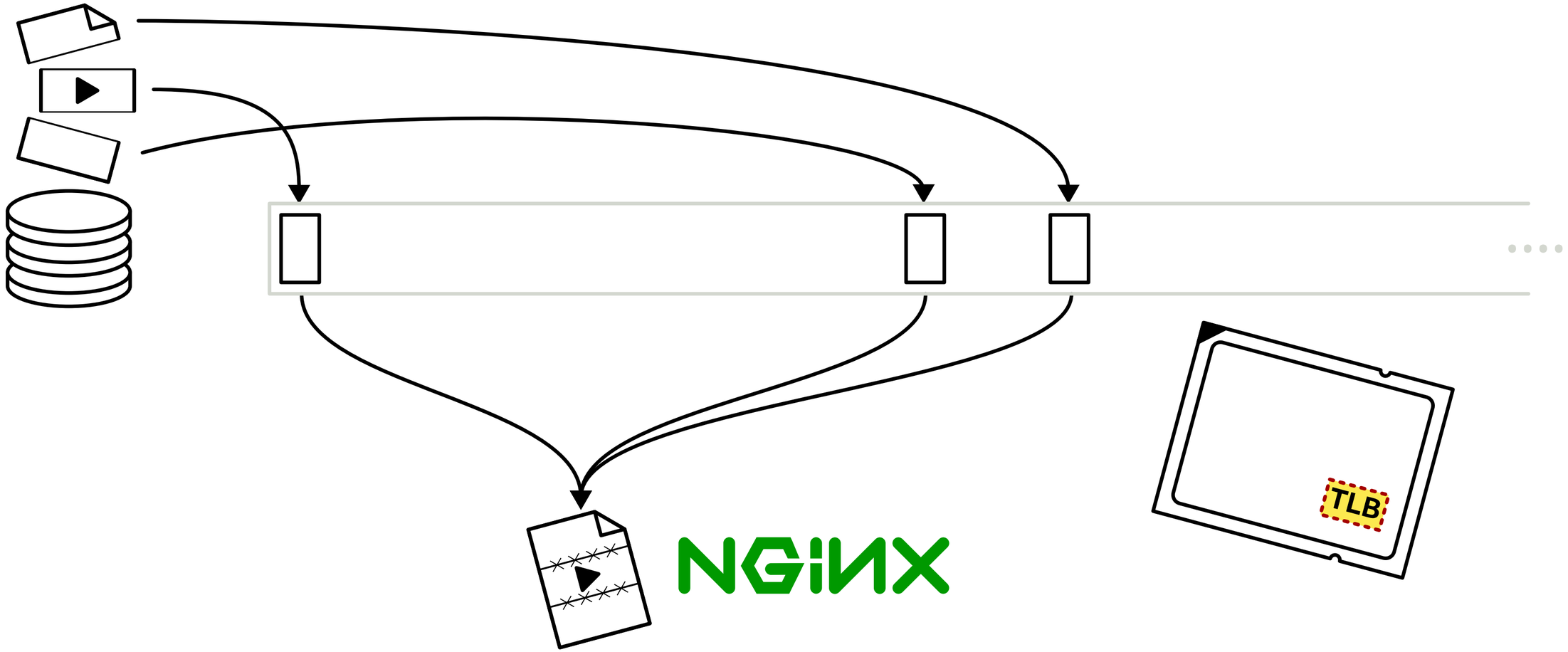
NGINX

Operating System Page Cache Primer



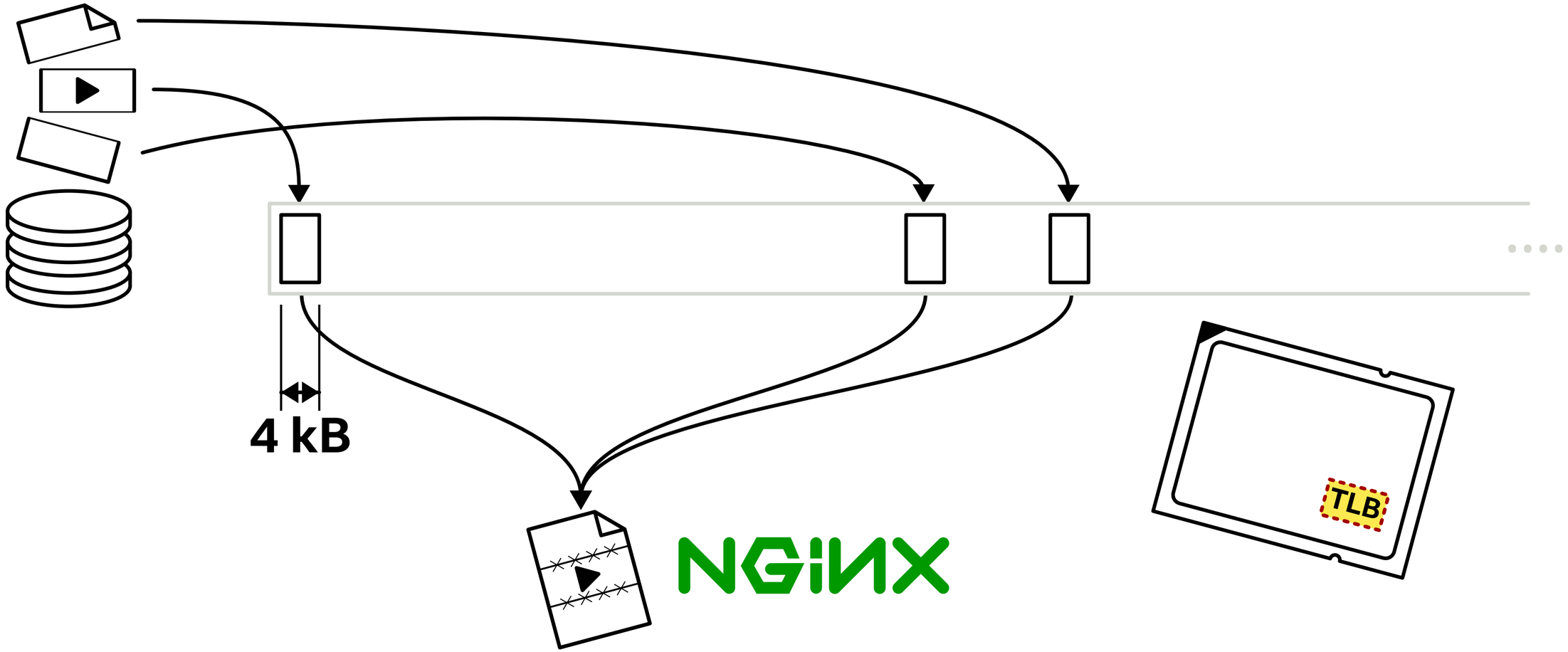
BBC

Operating System Page Cache Primer



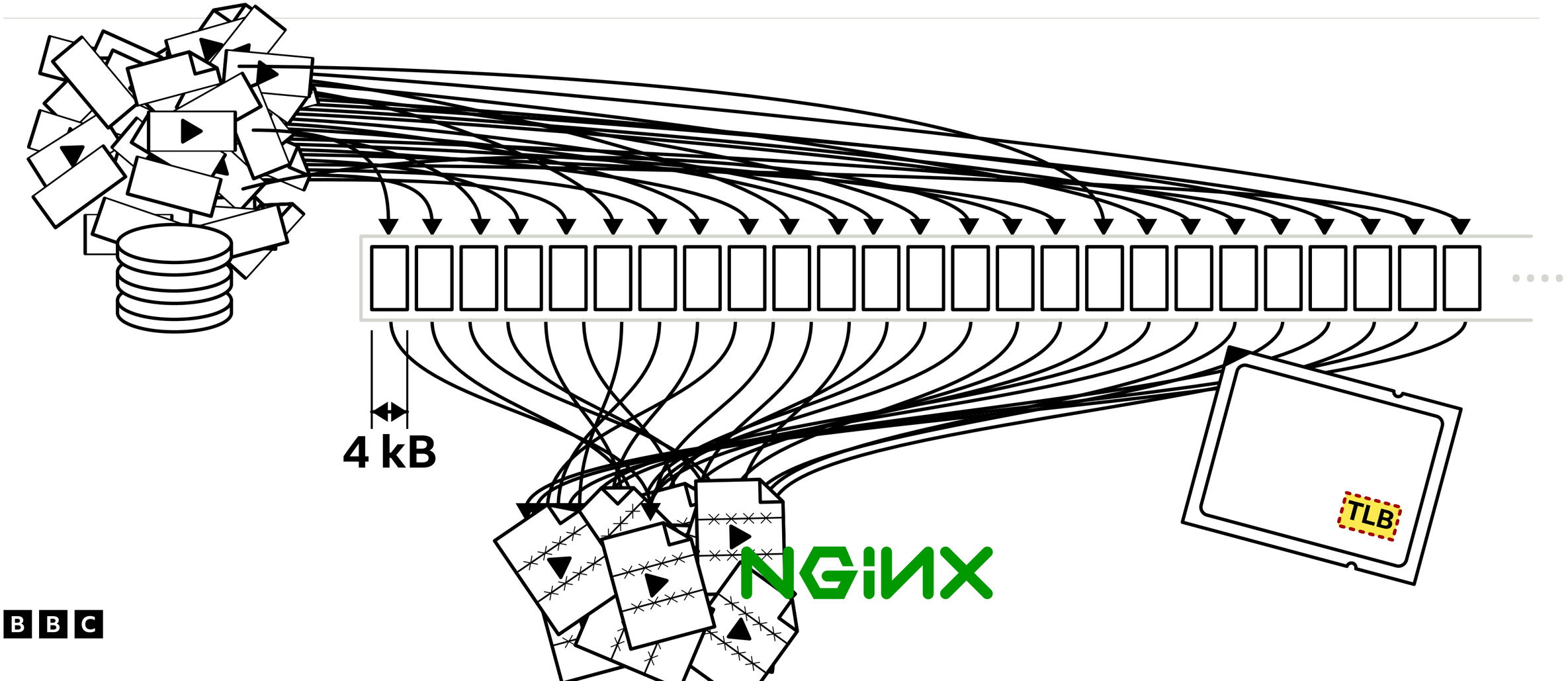
BBC

Operating System Page Cache Primer



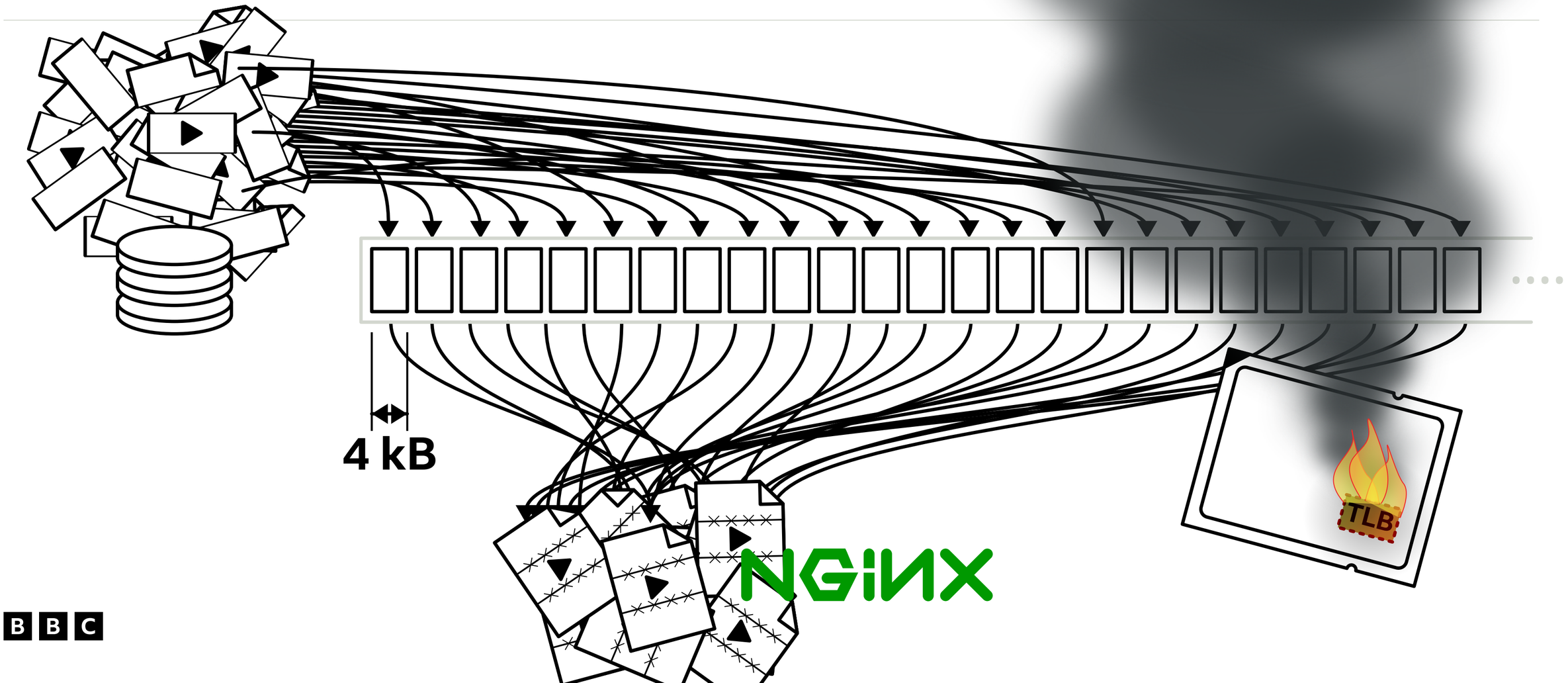
BBC

Operating System Page Cache Primer



BBC

Operating System Page Cache Primer



BBC

Available Page Sizes

4 kB
2 MB
1 GB

x86_64 (Intel/AMD)

BBC

Available Page Sizes

4 kB

~~2 MB~~

~~1 GB~~

x86_64 (Intel/AMD)

BBC

Available Page Sizes

x86_64 (Intel/AMD)	AARCH64 (ARM)
4 kB	4 kB
~~2 MB~~	16 kB
~~1 GB~~	64 kB
	2 MB
	32 MB
	512 MB
	1 GB

BBC

Available Page Sizes

4 kB

~~2 MB~~

~~1 GB~~

x86_64 (Intel/AMD)

4 kB 16 kB 64 kB

~~2 MB~~ ~~32 MB~~ ~~512 MB~~

~~1 GB~~

AARCH64 (ARM)

BBC

Available Page Sizes

4 kB

~~2 MB~~

~~1 GB~~

x86_64 (Intel/AMD)

4 kB 16 kB 64 kB

~~2 MB~~ ~~32 MB~~ ~~512 MB~~

~~1 GB~~

AARCH64 (ARM)

BBC

Available Page Sizes

4 kB

~~2 MB~~

~~1 GB~~

x86_64 (Intel/AMD)

4 kB 16 kB 64 kB

~~2 MB~~ ~~32 MB~~ ~~512 MB~~

~~1 GB~~

AARCH64 (ARM)

BBC

4kB Pages

64kB Pages

80 GBit/s

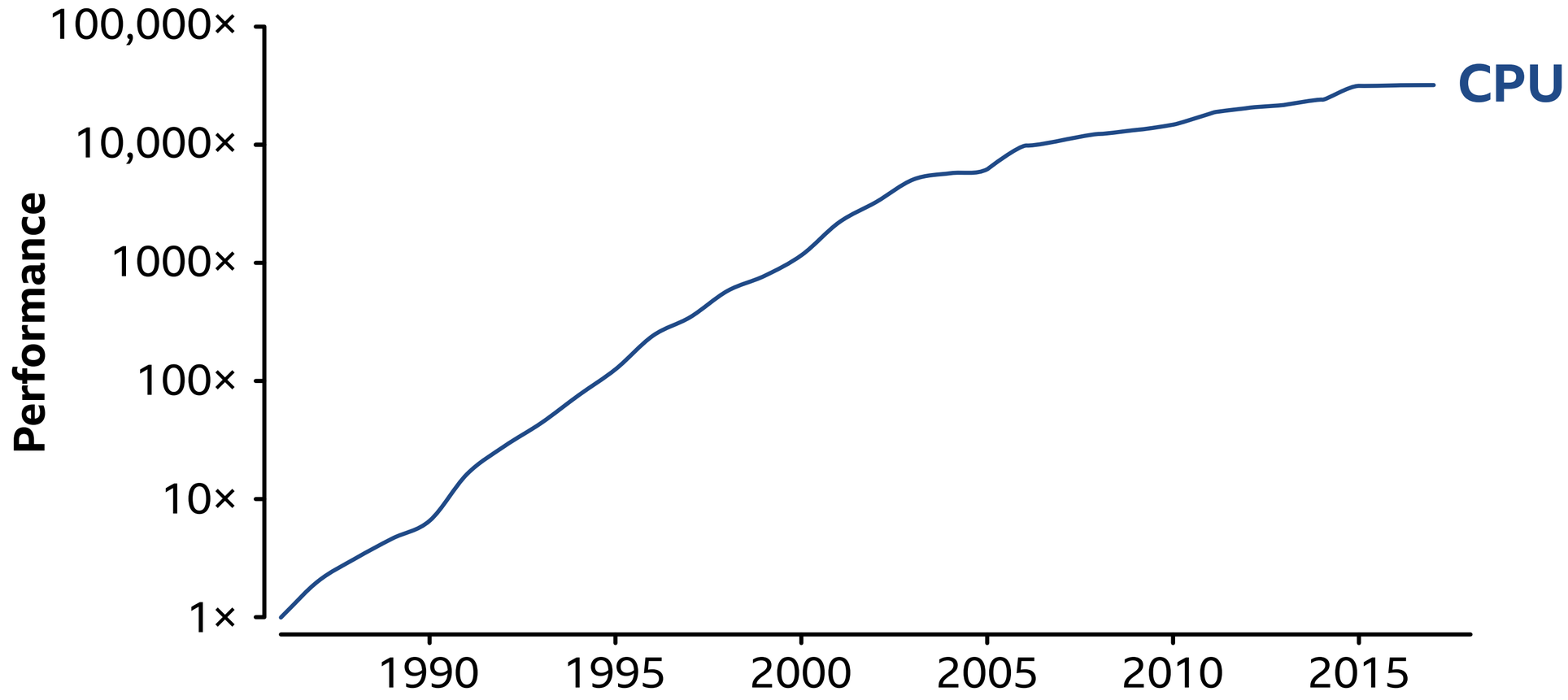
150 GBit/s

BBC

Memory Bandwidth

BBC

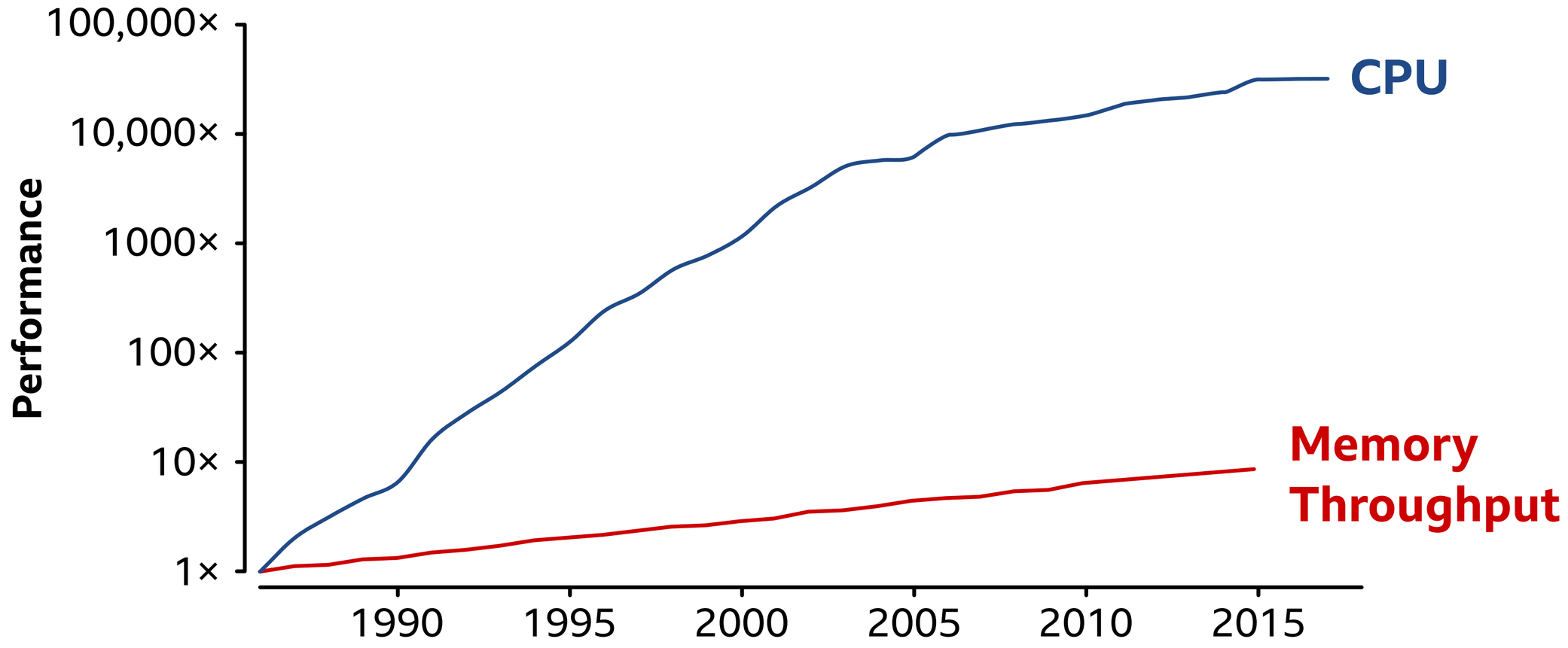
Bottleneck: Not the CPU anymore



BBC

Data from “Computer Architecture, A Quantitative Approach (Sixth Edition)”
Hennesy and Patterson 2019, figures 1.1 (CPU) and 2.2 (Memory Throughput).

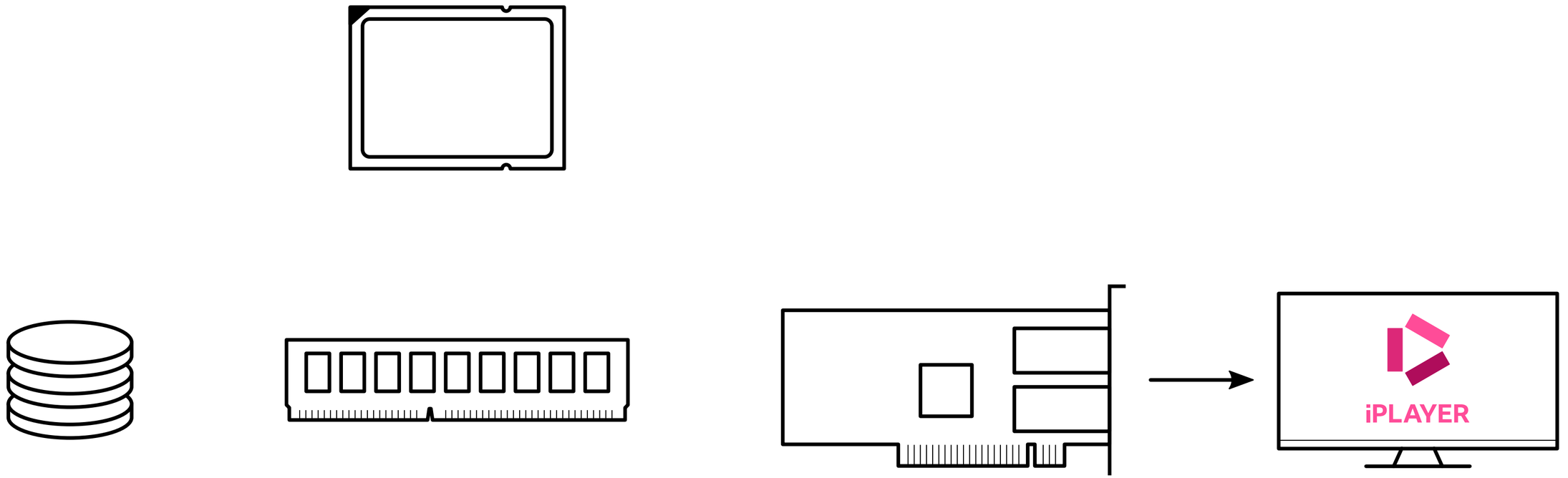
Bottleneck: Not the CPU anymore



BBC

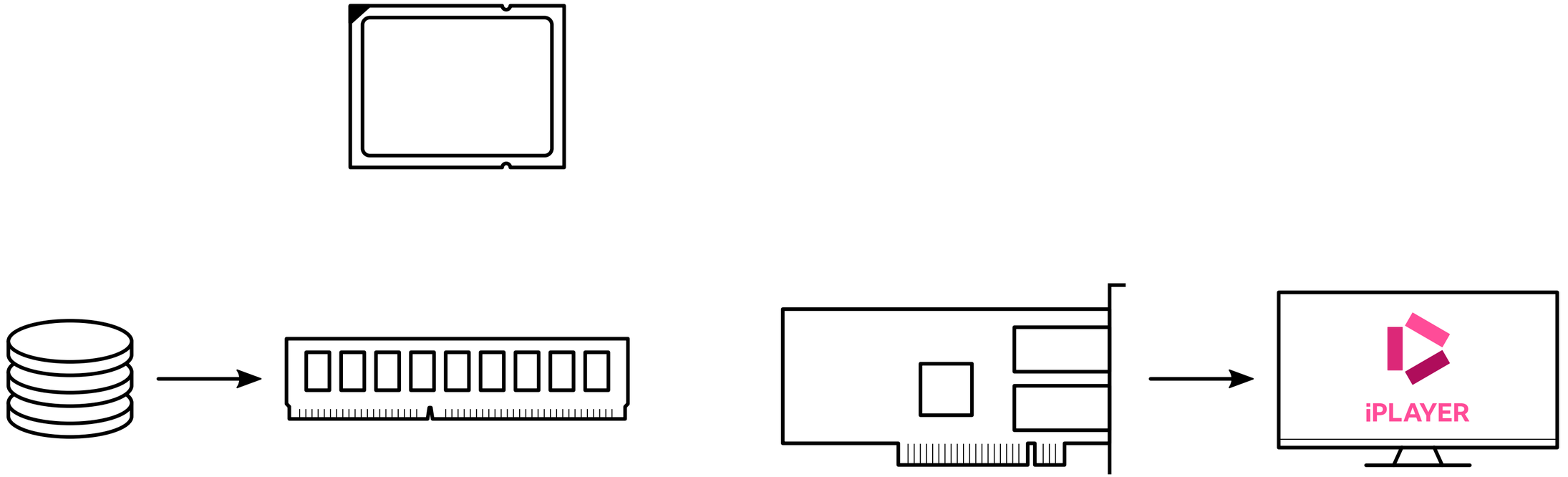
Data from “Computer Architecture, A Quantitative Approach (Sixth Edition)”
Hennesy and Patterson 2019, figures 1.1 (CPU) and 2.2 (Memory Throughput).

Where is the memory bandwidth going?



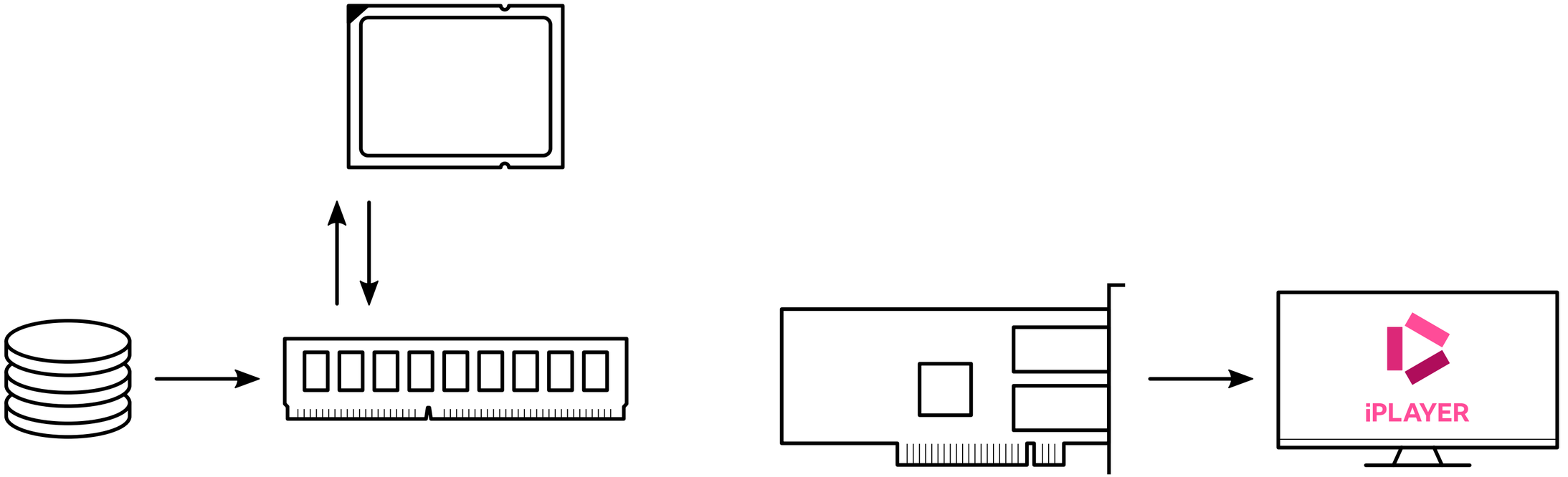
BBC

Where is the memory bandwidth going?



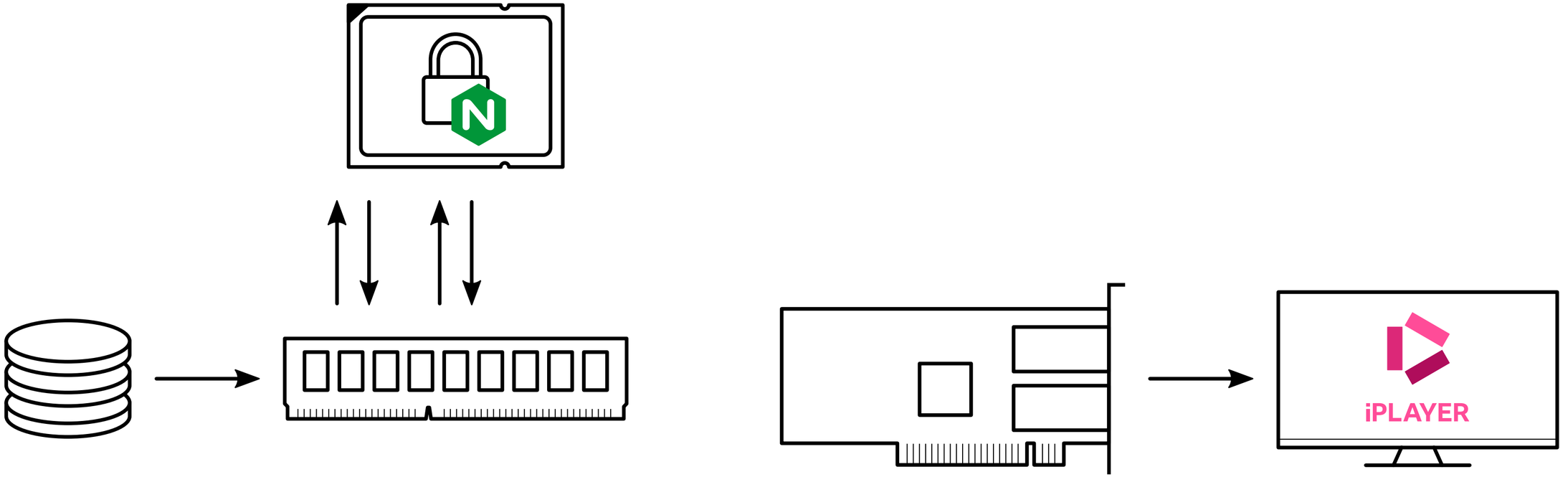
BBC

Where is the memory bandwidth going?



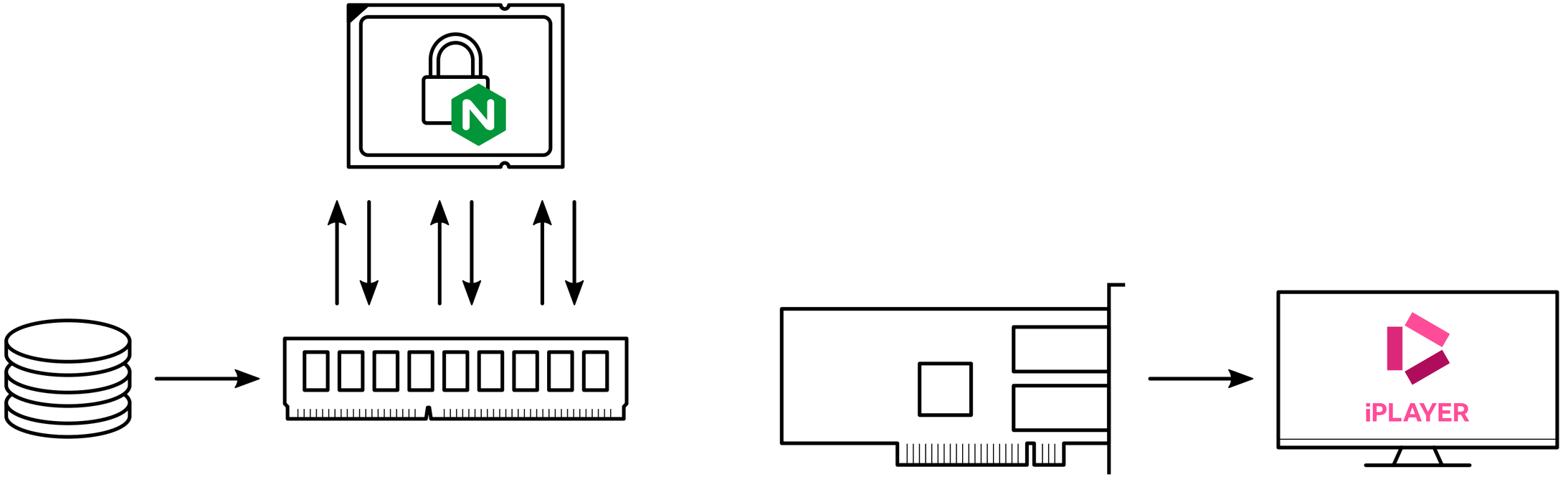
BBC

Where is the memory bandwidth going?



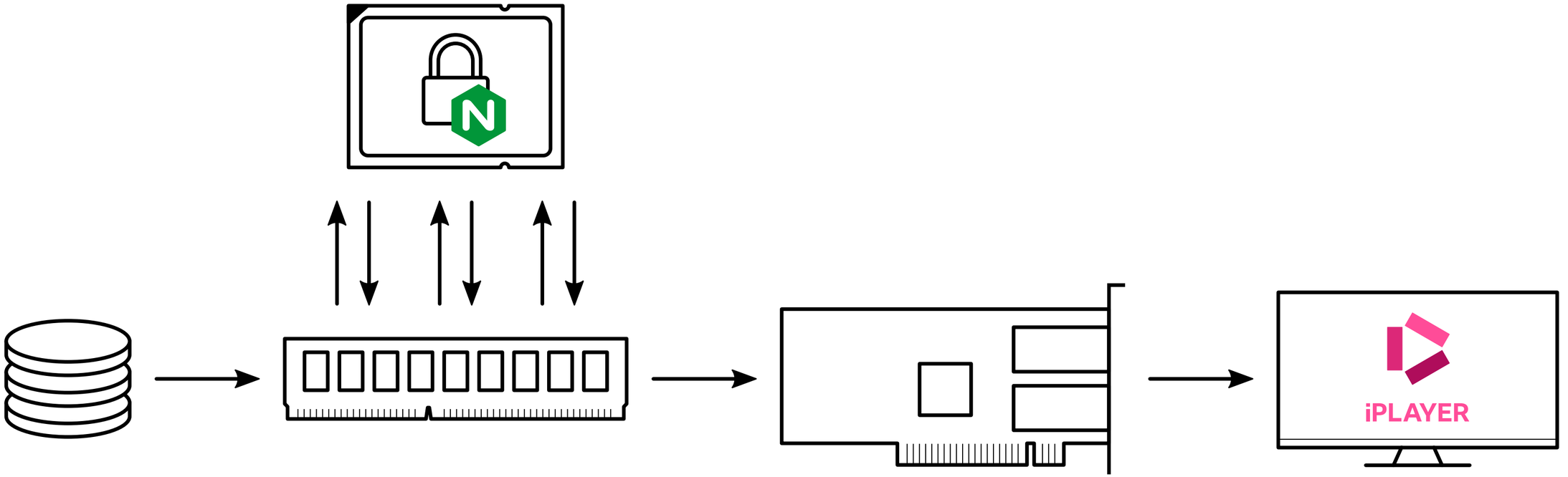
BBC

Where is the memory bandwidth going?



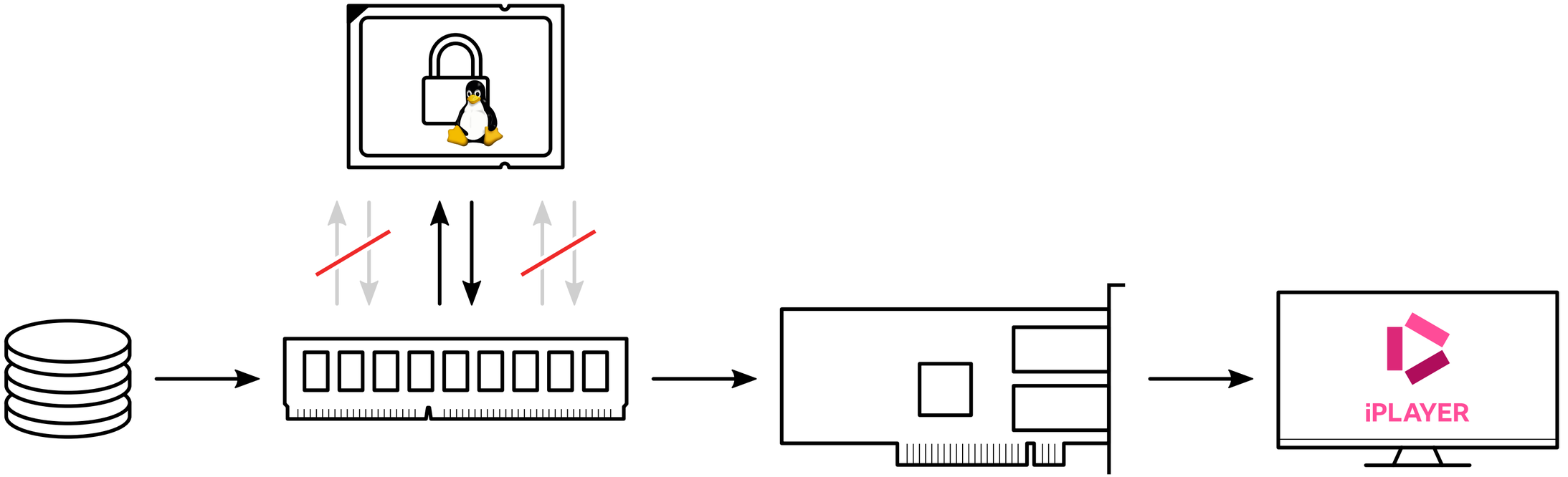
BBC

Where is the memory bandwidth going?



BBC

Kernel TLS (kTLS)



BBC

Userspace TLS

Kernel TLS

150 GBit/s

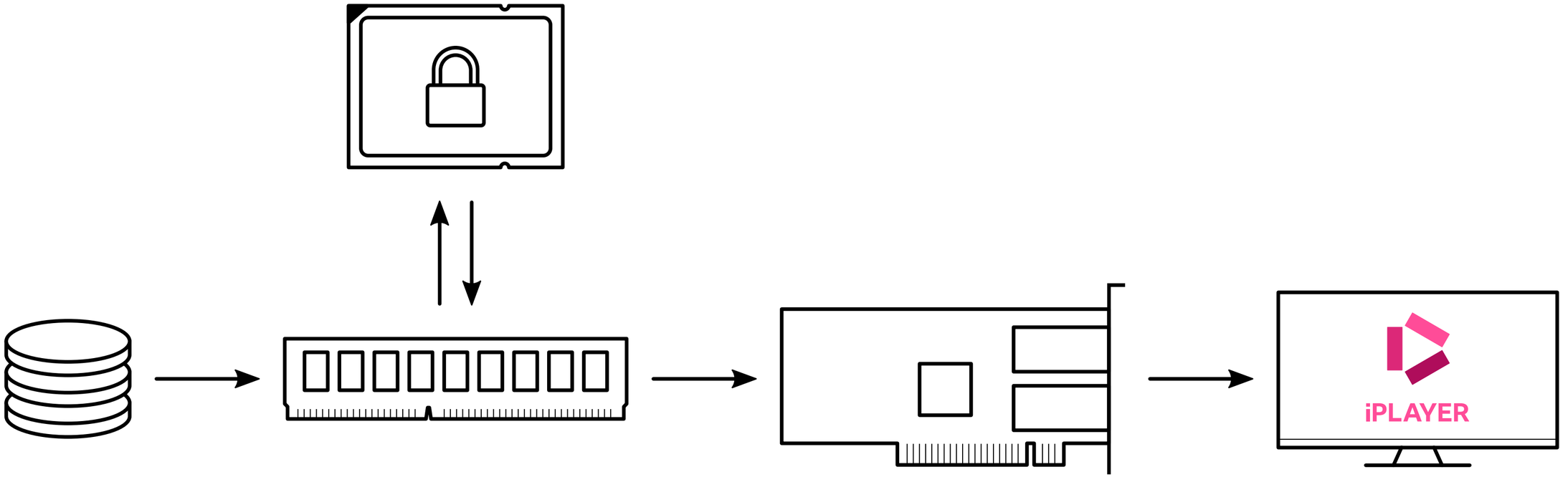
260 GBit/s

BBC

TLS Offload

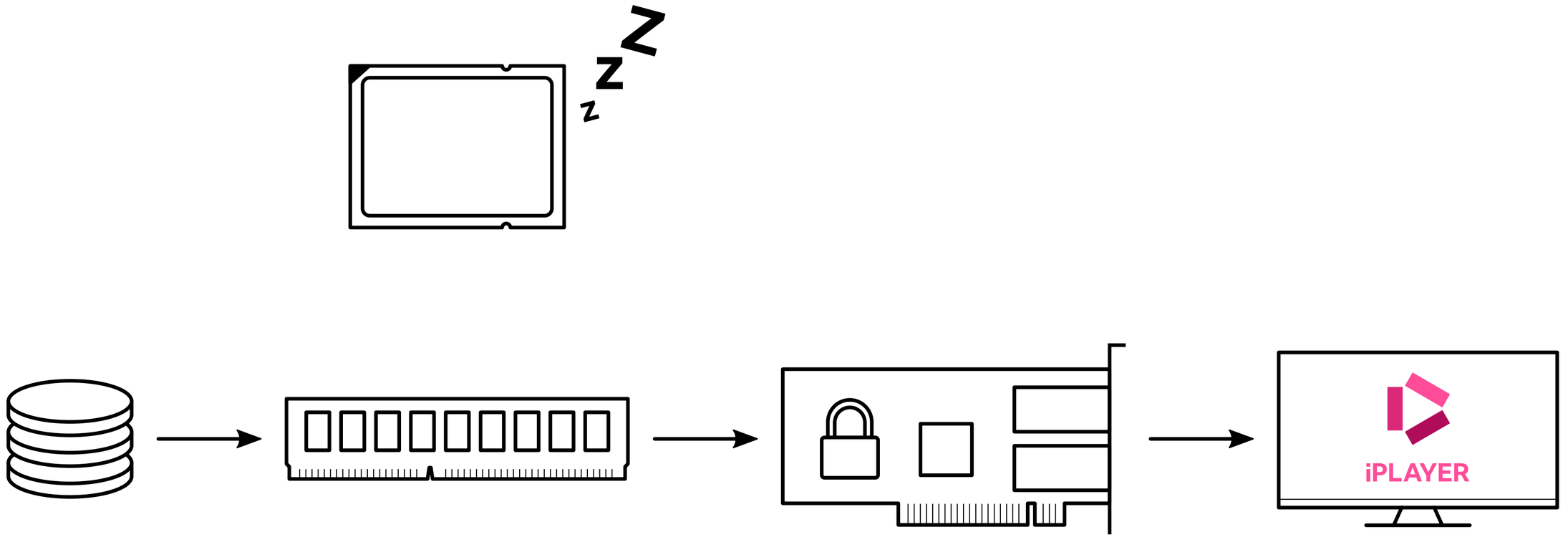
BBC

TLS Offload: The theory



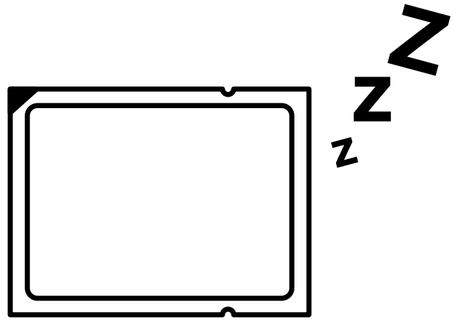
BBC

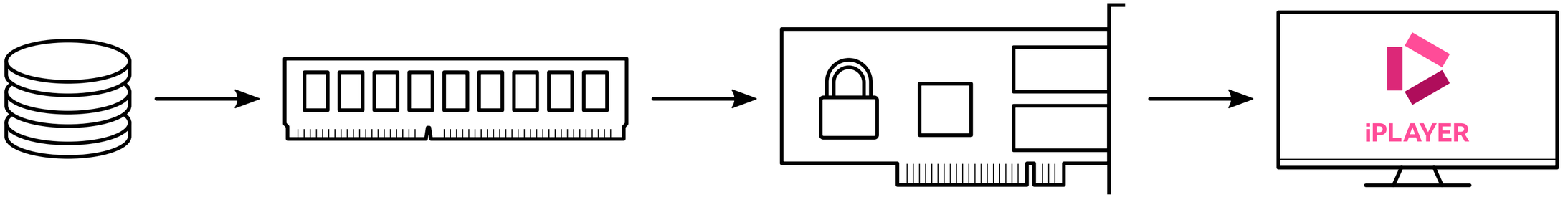
TLS Offload: The theory



BBC

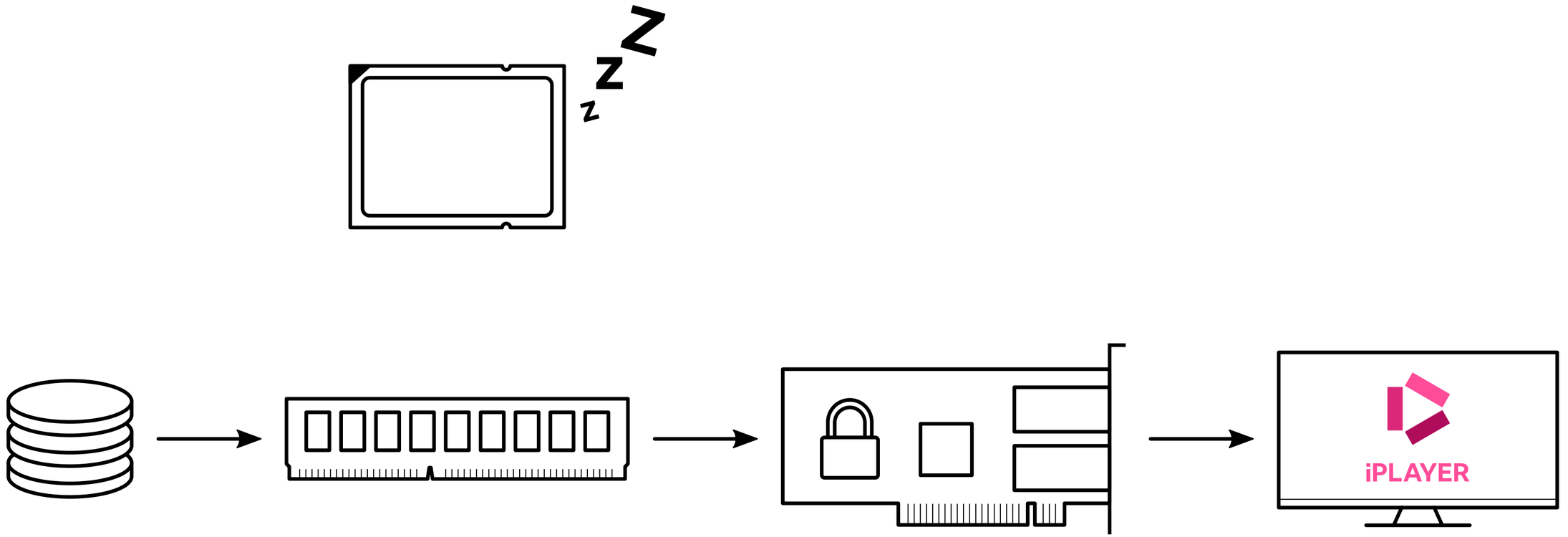
TLS Offload: The theory





BBC

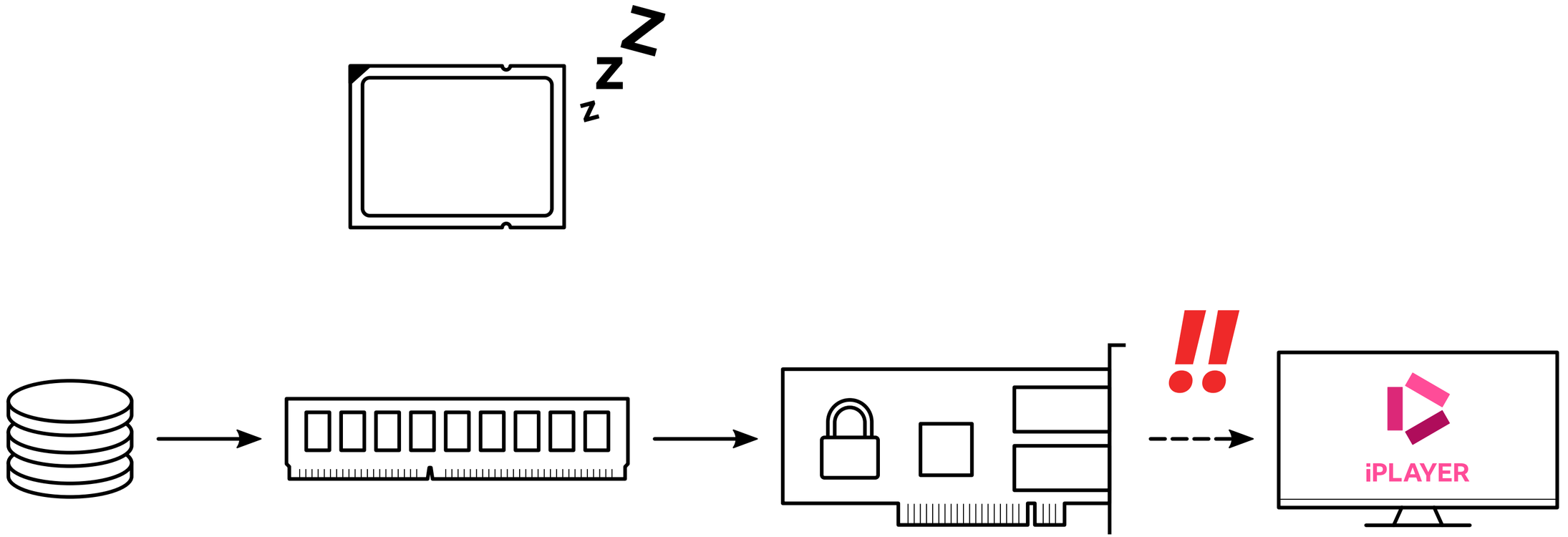
TLS Offload: The reality



BBC

TLS Offload: The reality

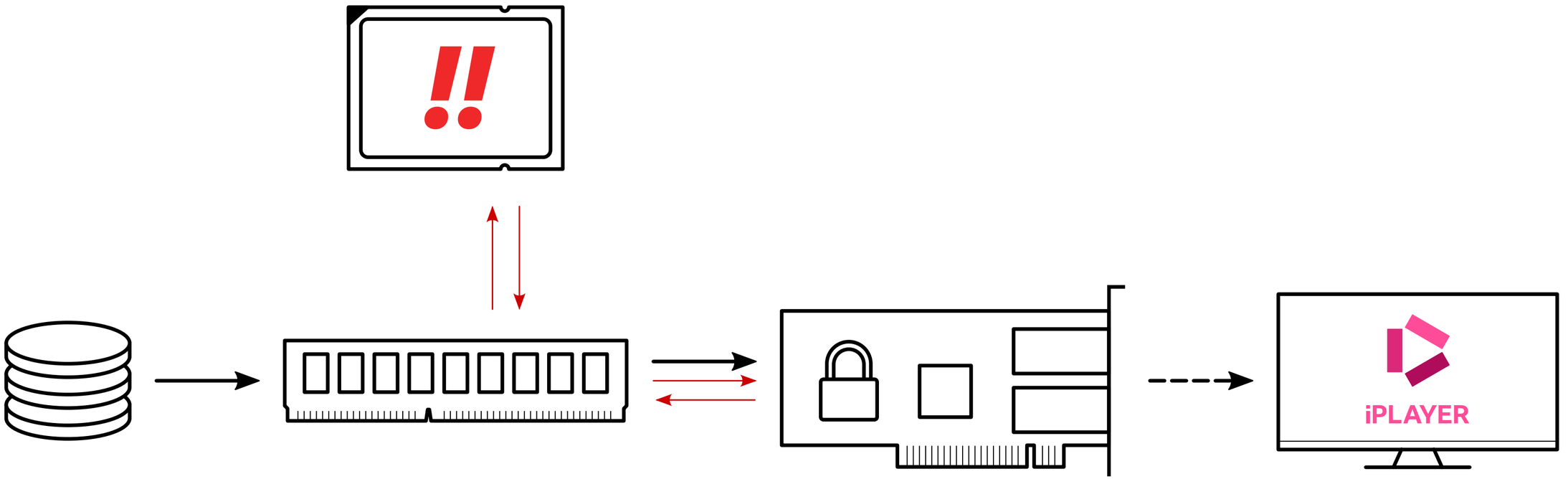
Problem 1: Retransmissions



BBC

TLS Offload: The reality

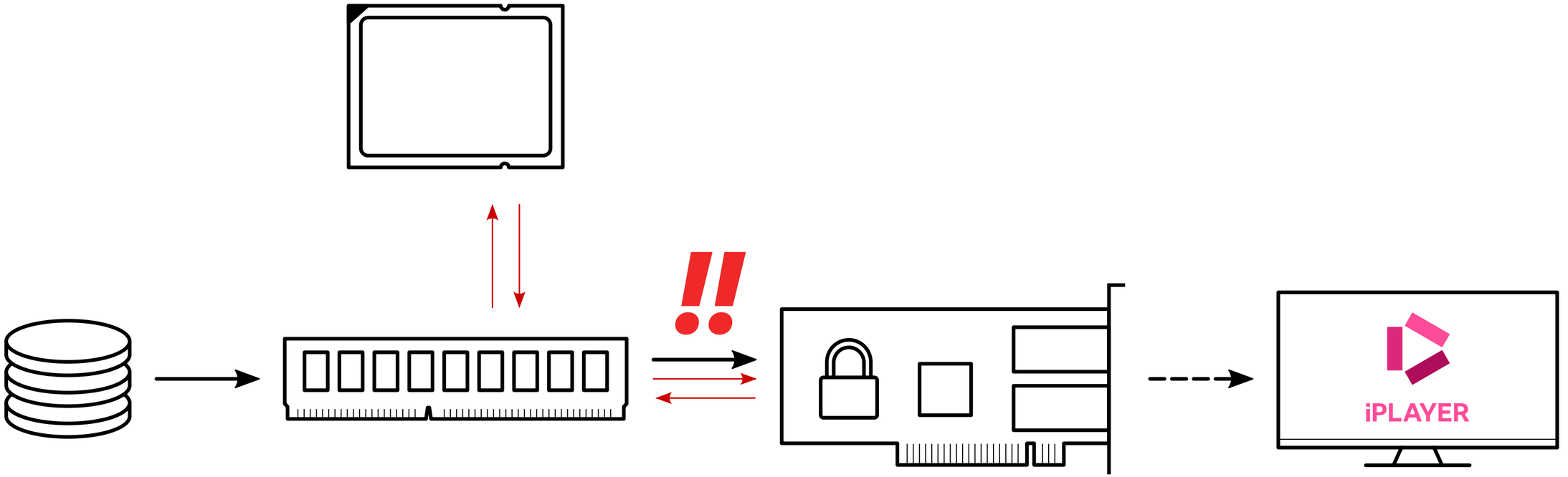
Problem 1: Retransmissions



BBC

TLS Offload: The reality

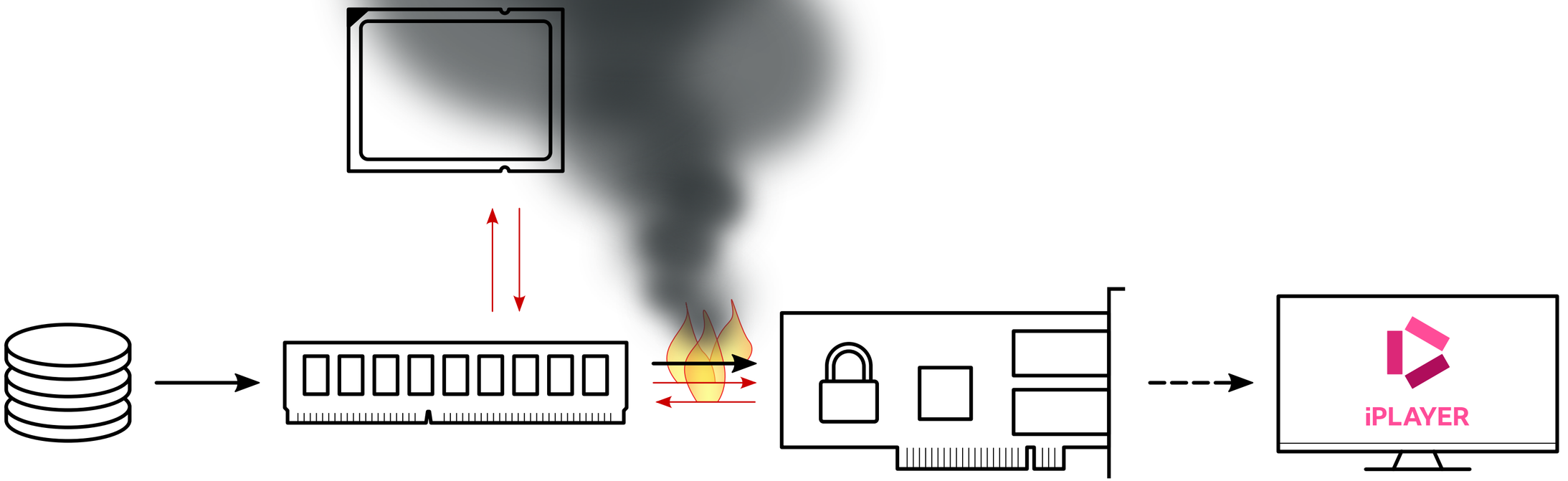
Problem 1: Retransmissions



BBC

TLS Offload: The reality

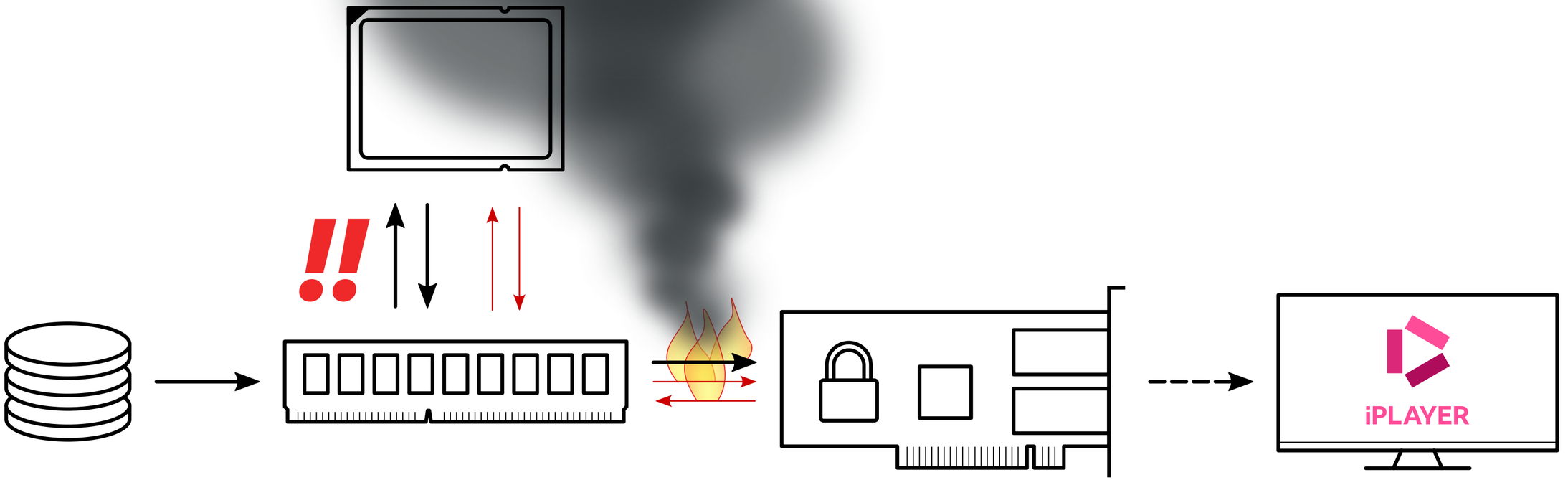
Problem 1: Retransmissions



BBC

TLS Offload: The reality

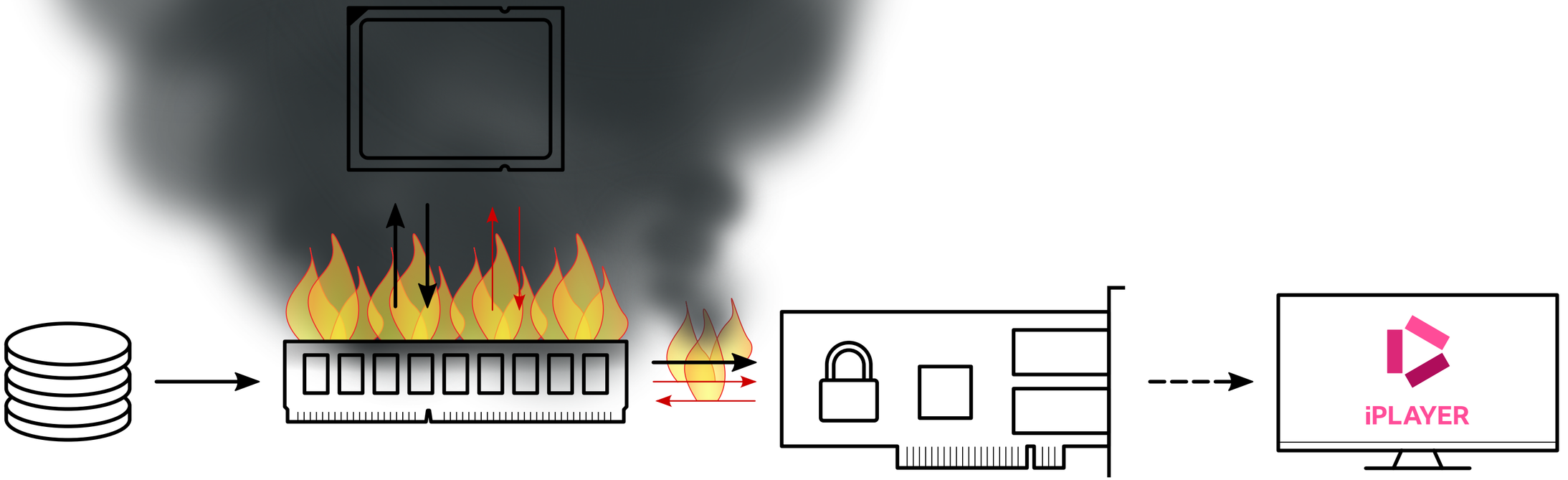
Problem 1: Retransmissions
Problem 2: Linux implementation



BBC

TLS Offload: The reality

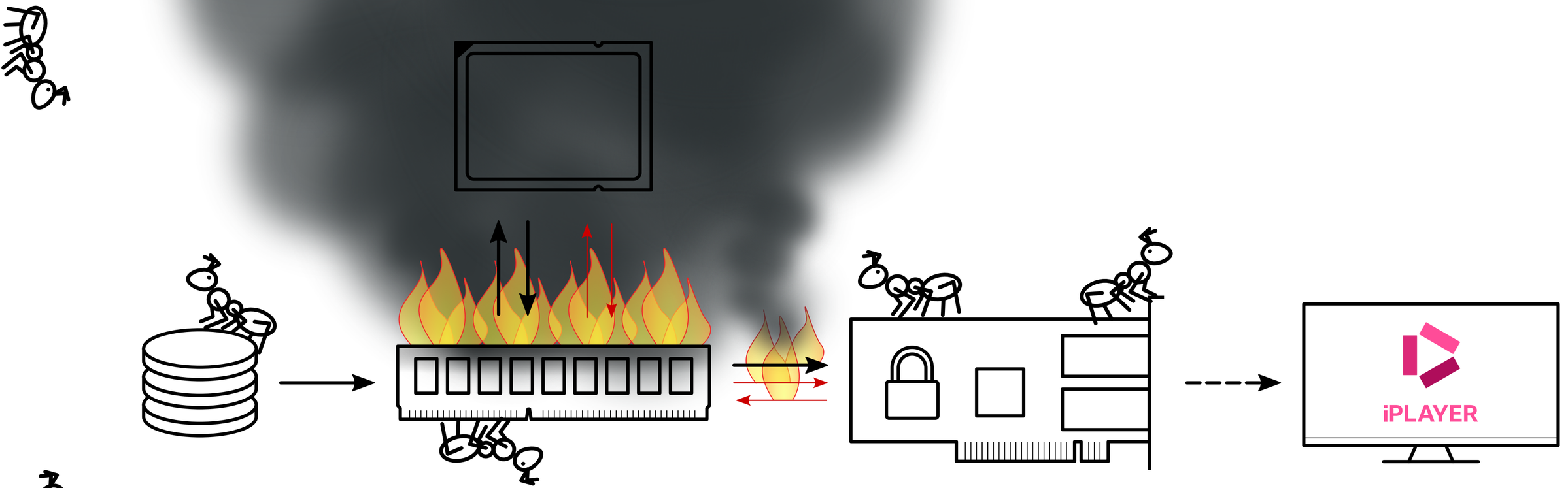
Problem 1: Retransmissions
Problem 2: Linux implementation



BBC

TLS Offload: The reality

Problem 1: Retransmissions
Problem 2: Linux implementation
Problem 3: Stability

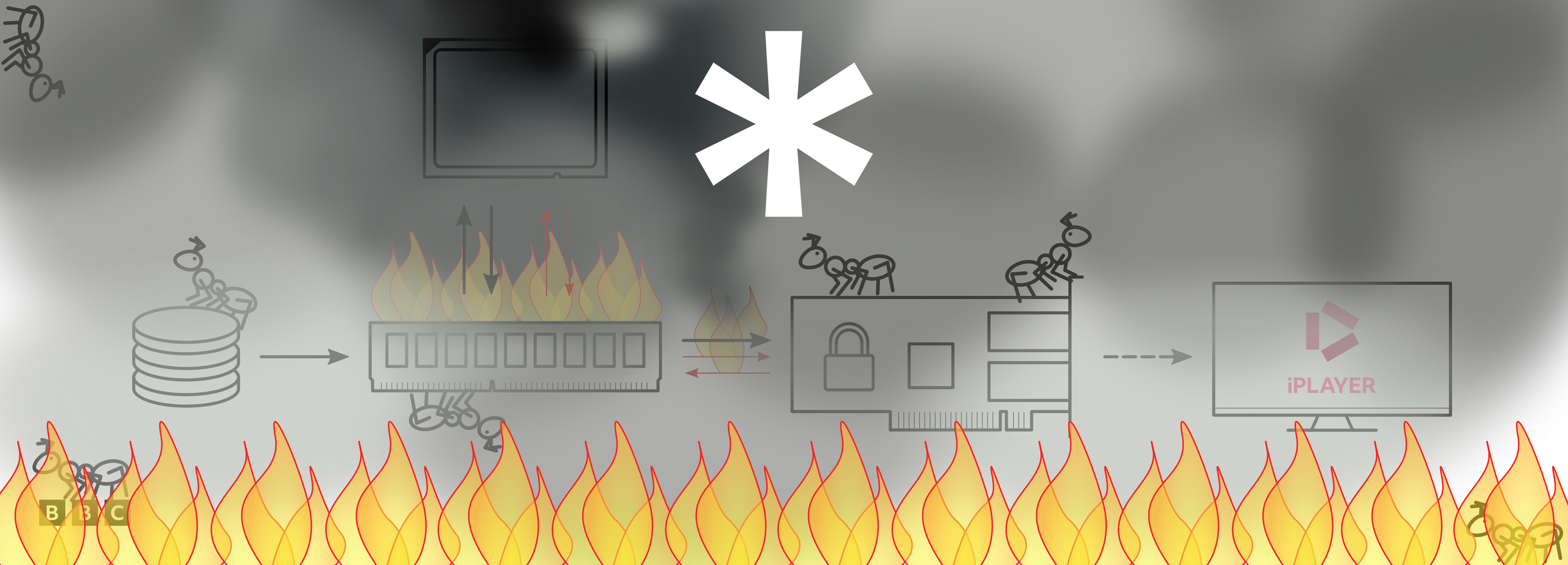


TLS Offload: The reality

Problem 1: Retransmissions
Problem 2: Linux implementation
Problem 3: Stability

TLS Offload: The reality

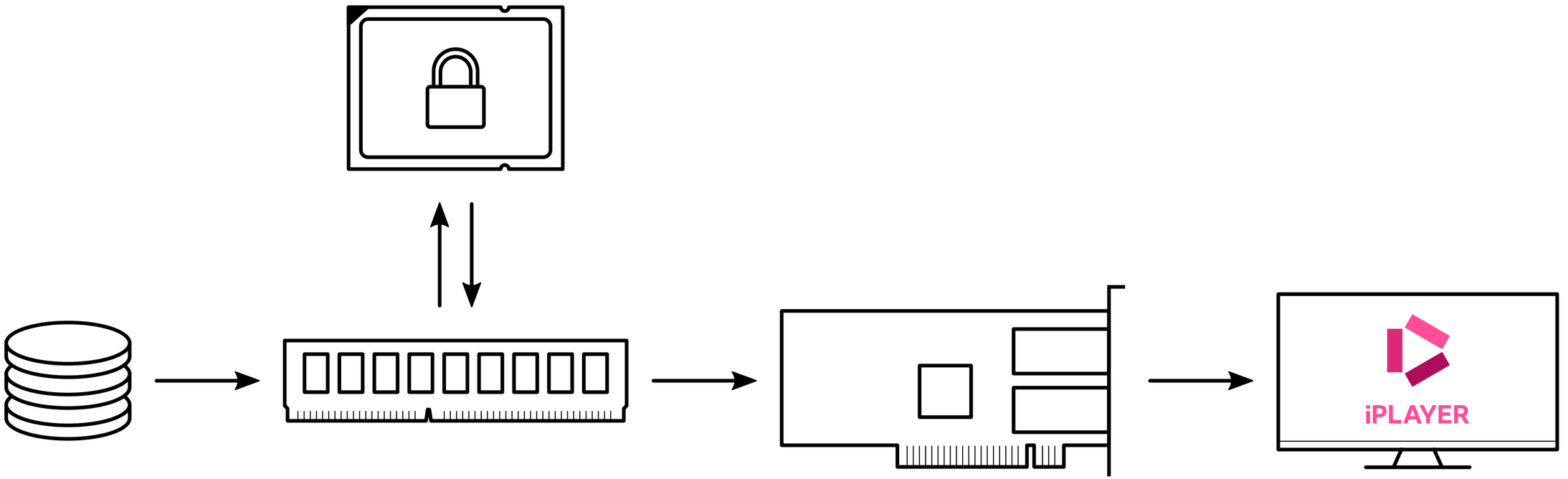
Problem 1: Retransmissions
Problem 2: Linux implementation
Problem 3: Stability



Network Behaviour

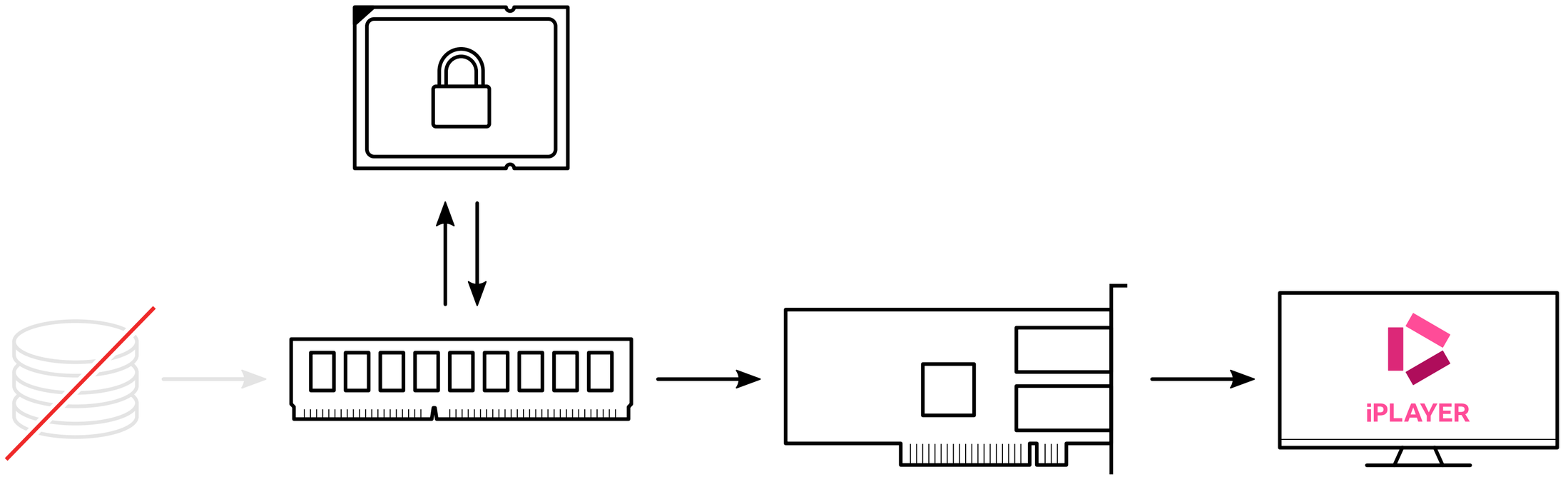
BBC

Request pattern effects



BBC

Request pattern effects



BBC

Served From Disk

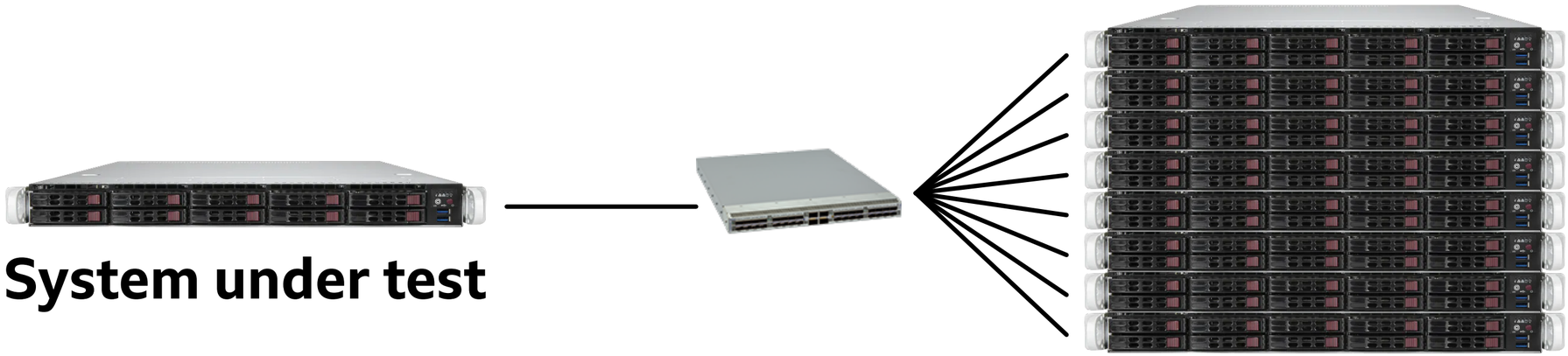
Served from RAM

260 GBit/s

340 GBit/s

BBC

Test lab realism



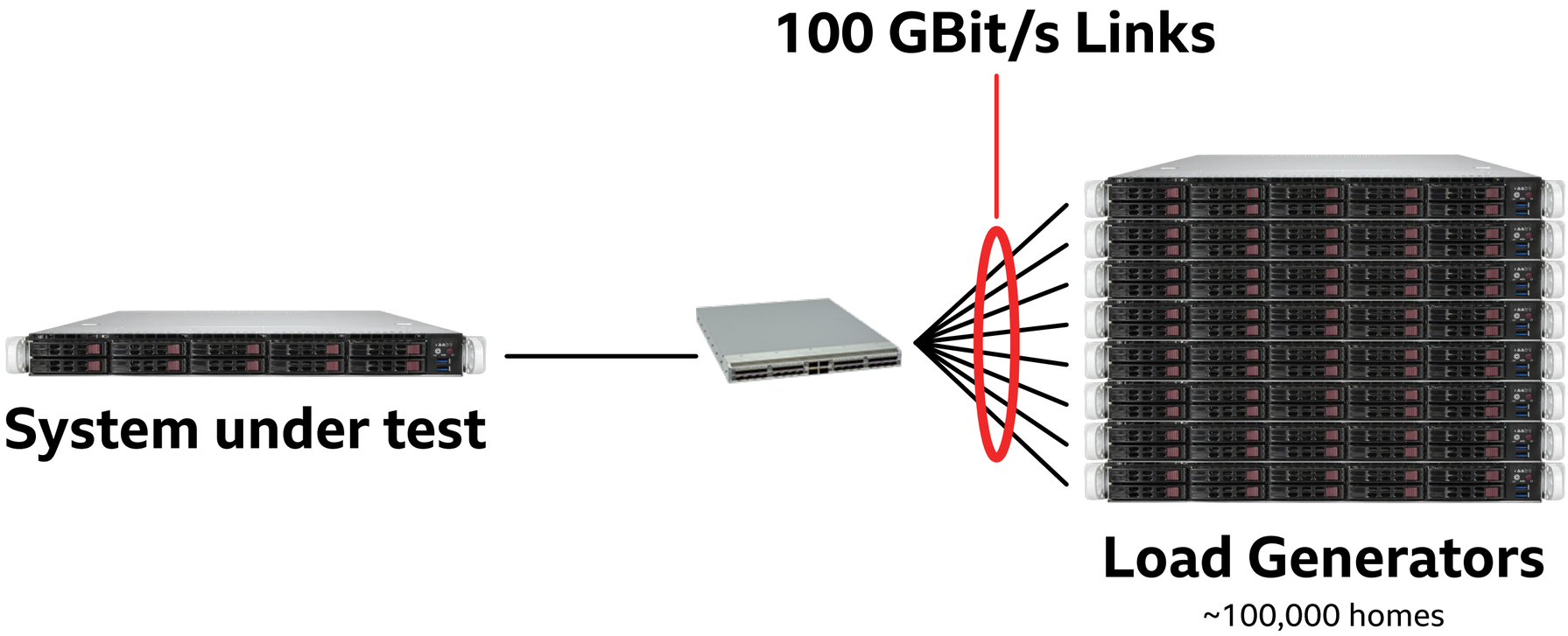
System under test

Load Generators

~100,000 homes

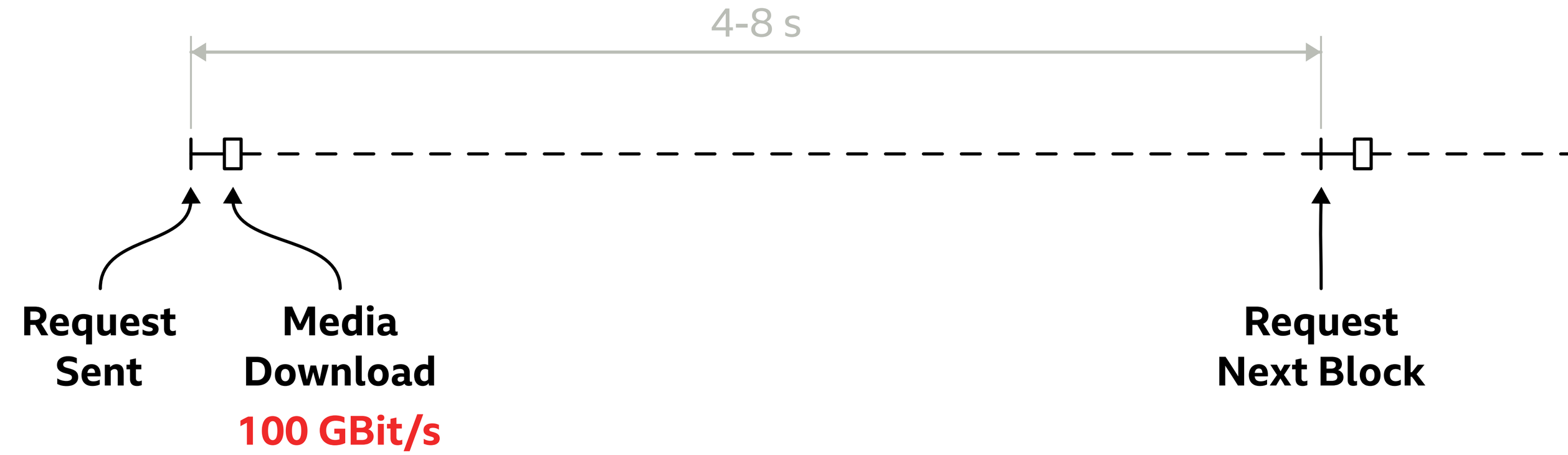
BBC

Test lab realism



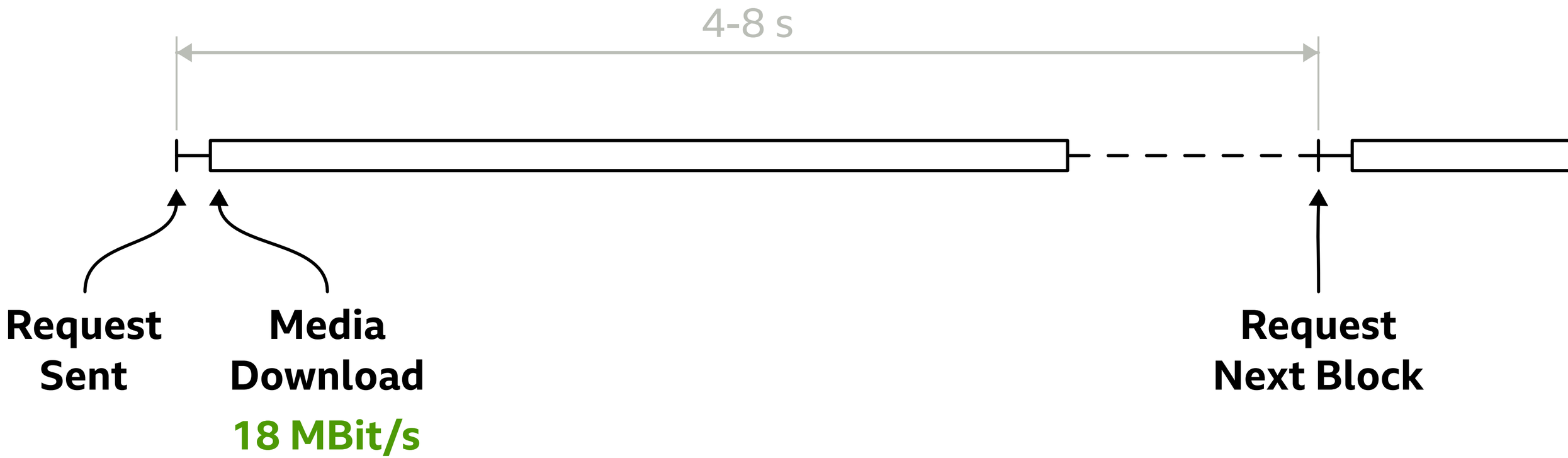
BBC

Client behaviour



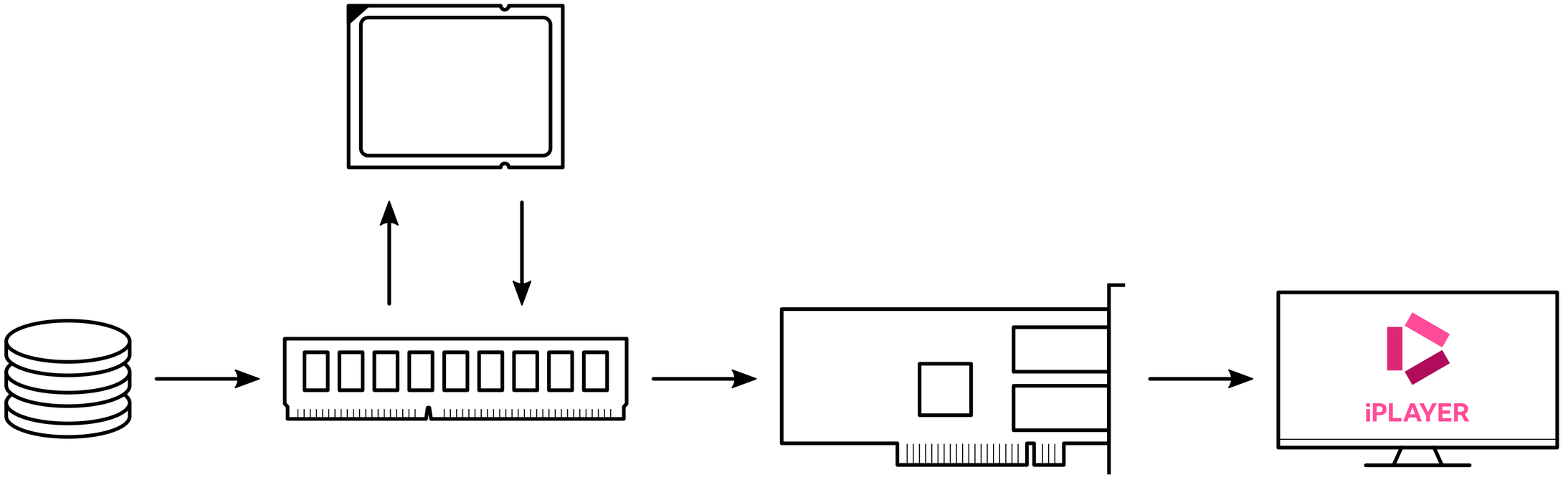
BBC

Client behaviour



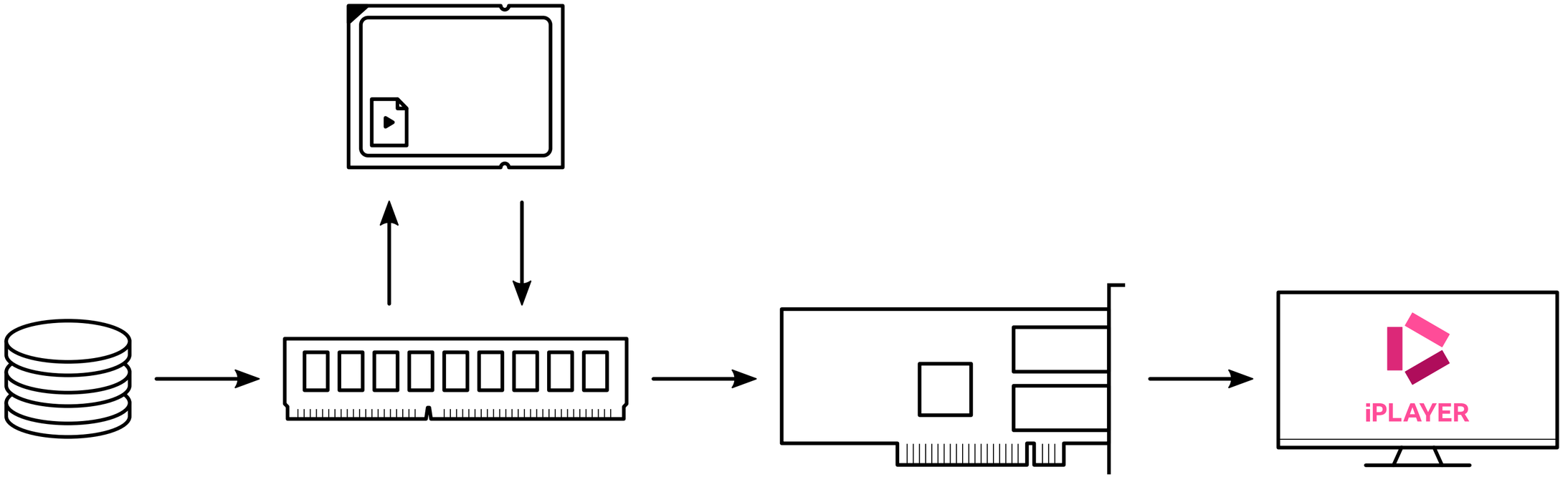
BBC

Slow client effects



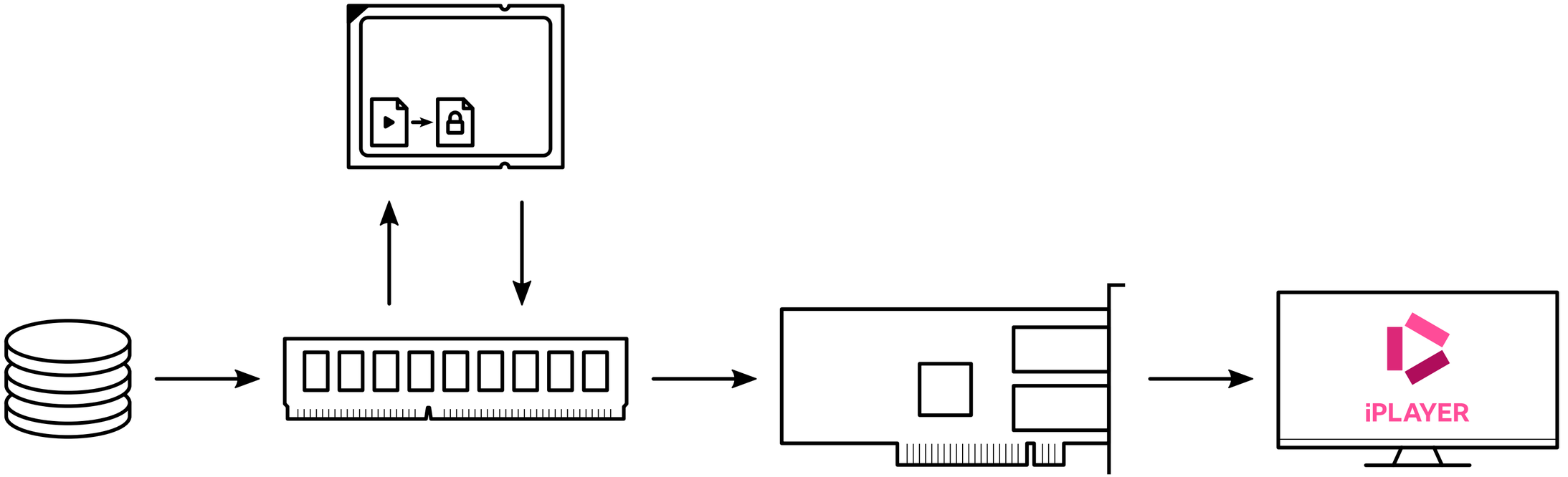
BBC

Slow client effects



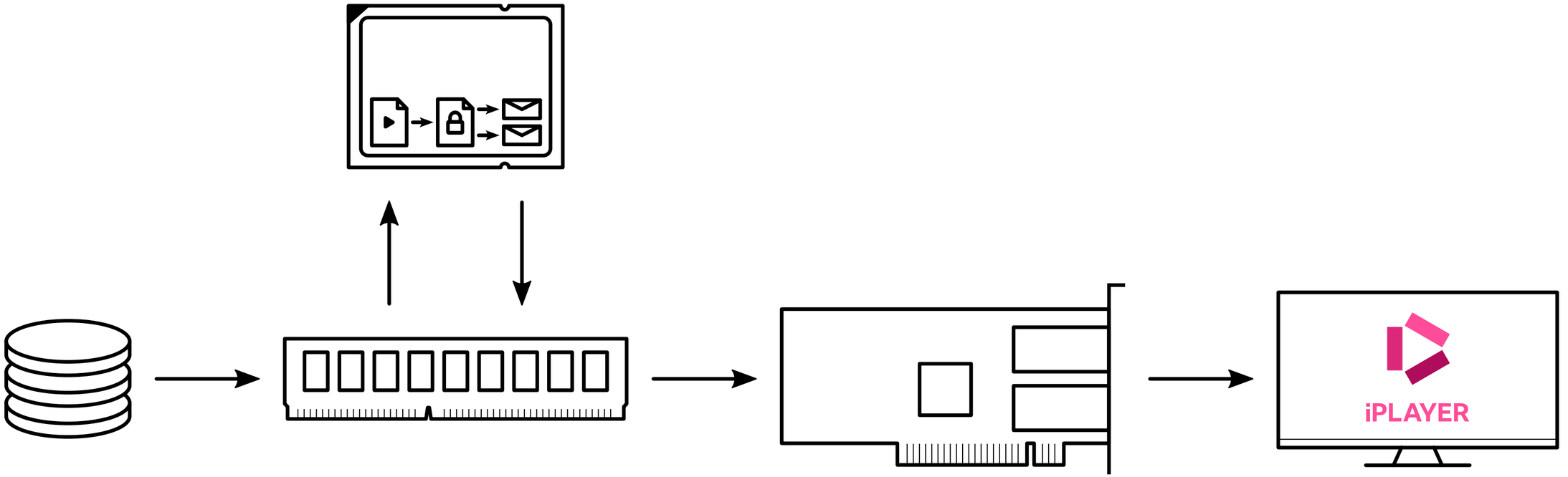
BBC

Slow client effects



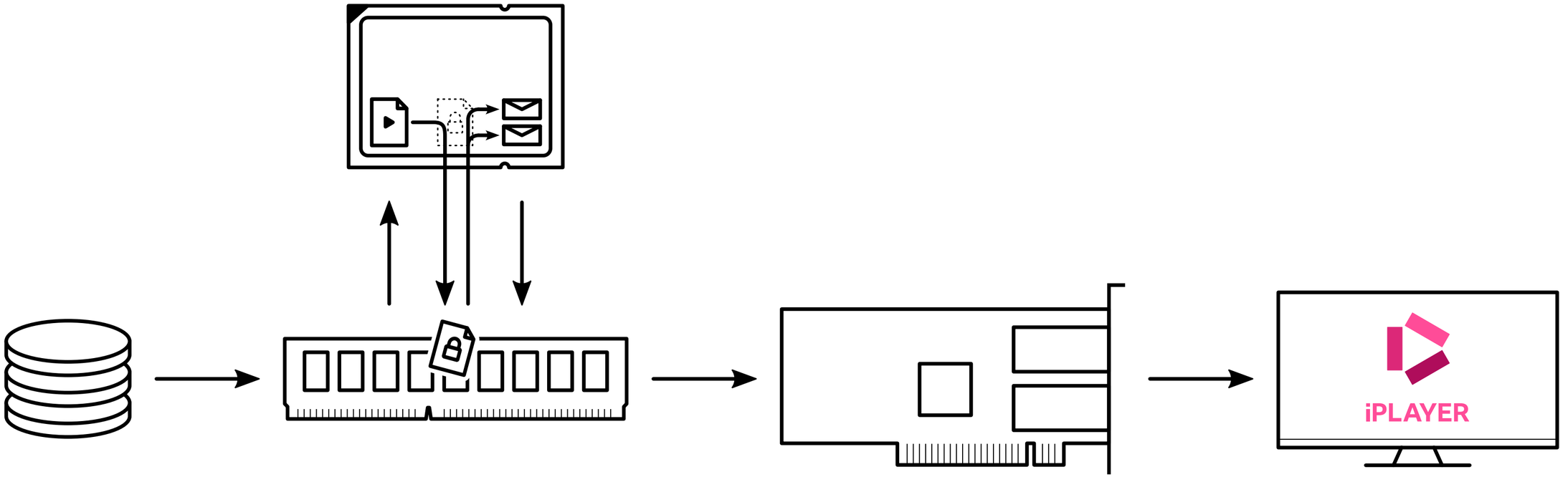
BBC

Slow client effects



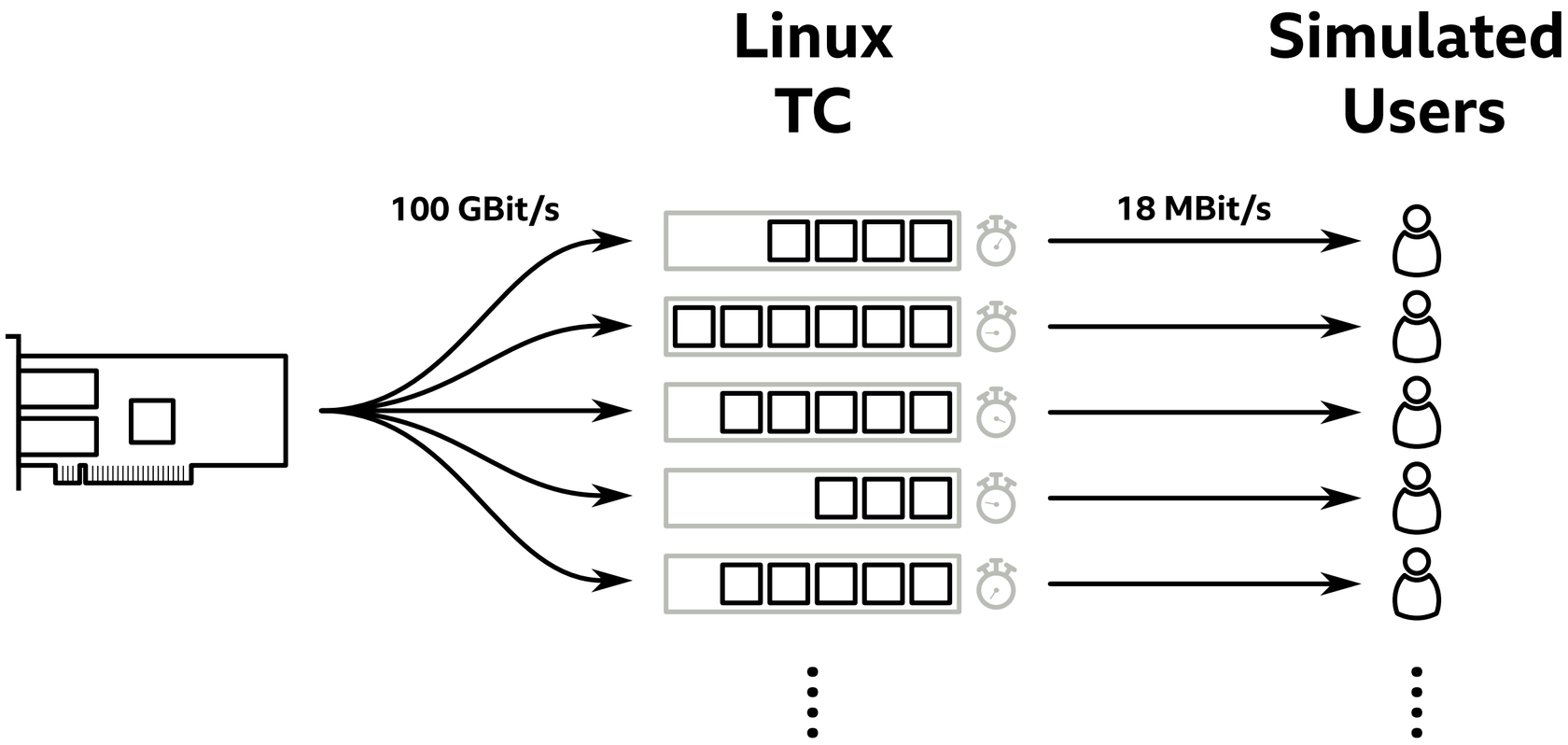
BBC

Slow client effects



BBC

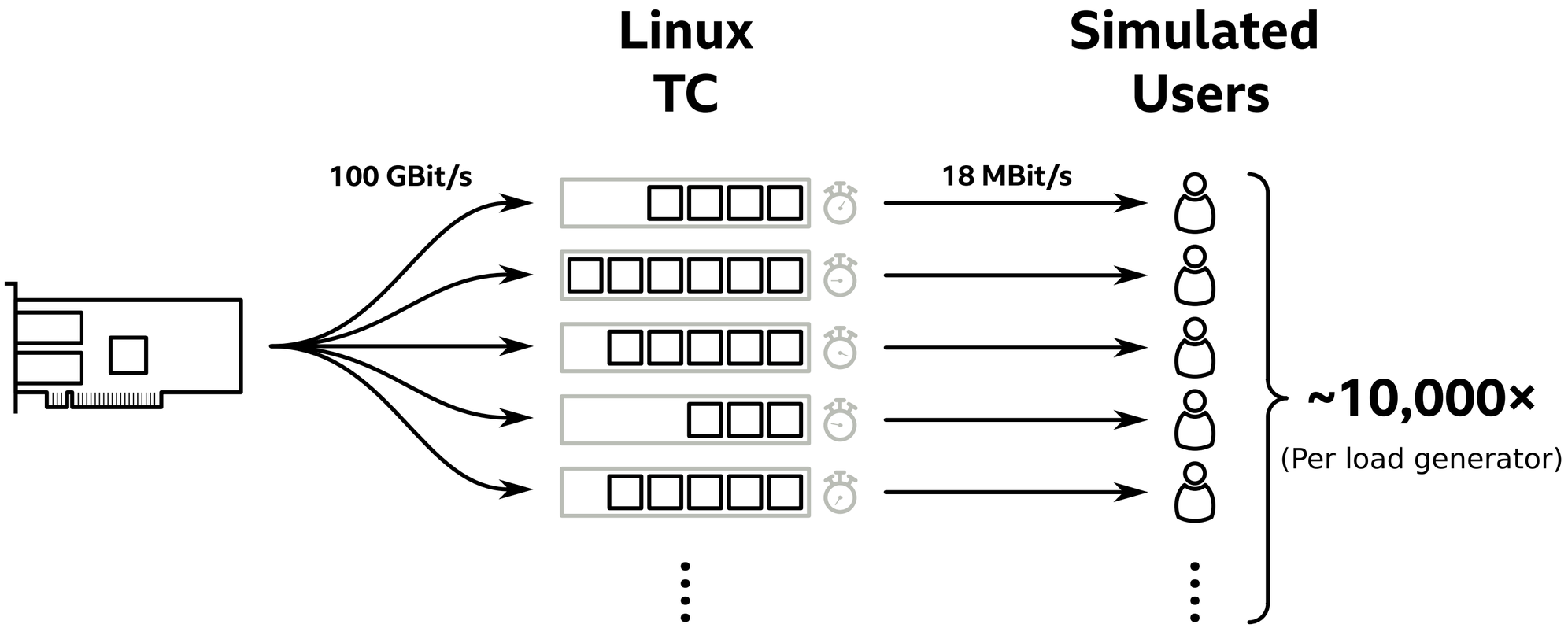
Modelling real internet connections: Linux TC



BBC

For a superb introduction TC see: https://medium.com/criteo-engineering/demystification-of-tc-de3dfe4067c2

Modelling real internet connections: Linux TC



BBC

For a superb introduction TC see: https://medium.com/criteo-engineering/demystification-of-tc-de3dfe4067c2

Modelling real internet connections: Linux TC

Sign up Sign in

Medium Search

Demystification of TC

Frederic Boismenu · Follow
Published in Criteo R&D Blog
15 min read · Nov 22, 2018

Listen Share

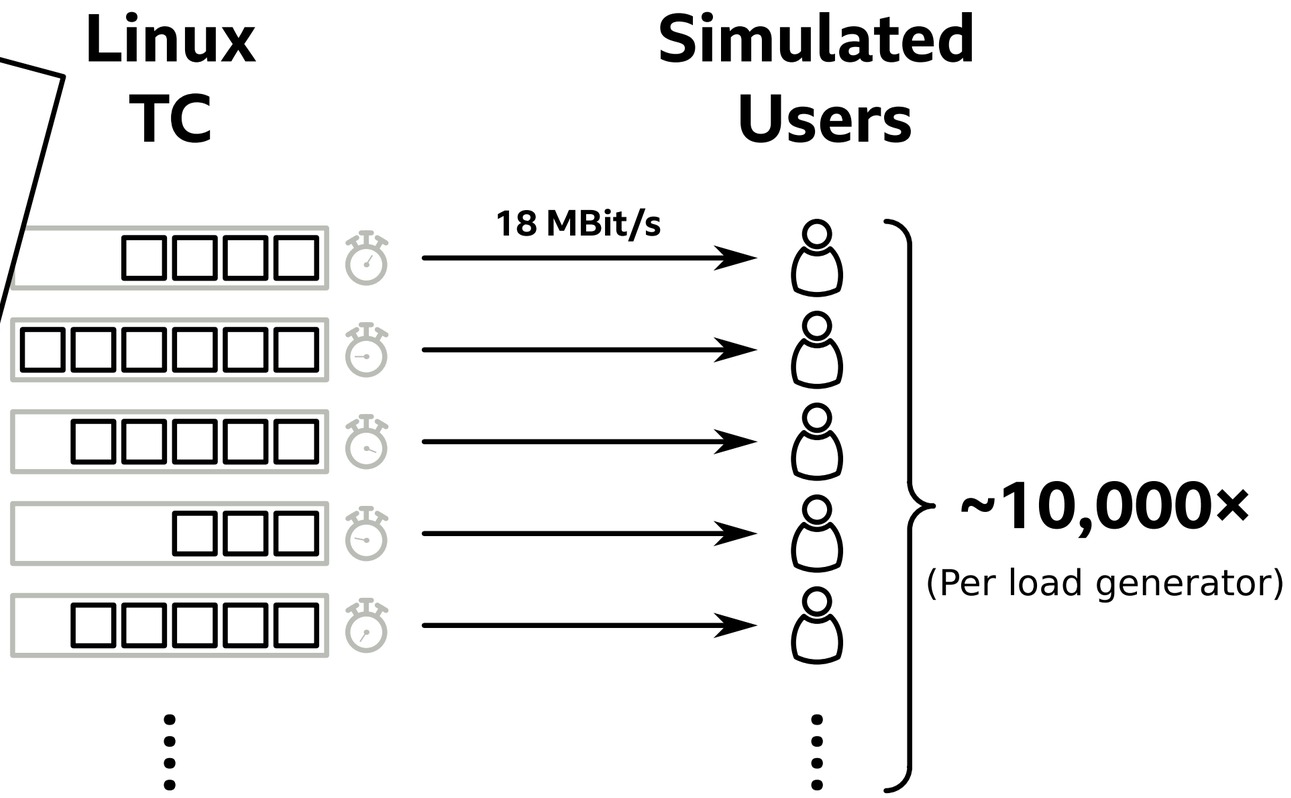
At Criteo we operate containers with Mesos. Containers are sharing the resources of their host. Applications are competing to access these resources. One of the great benefits of containers is to allow tackling the resource contention issue.

The network resources are somewhat special compared to other ones such as CPU and memory for several reasons. First of all, it is the most sensitive to contention just because of the (relative) lack of speed of the network compared to CPU and RAM. A network bound application will notice any added latency to its flow, which may be visible at human scale. It is easy for an application to impact the network performance of a co-hosted one. This is the noisy neighbor issue.

Then, network optimization issues are much more complex than, for example, CPU sharing because the overall performance of the system depends on external, unpredictable, and rapidly changing factors.

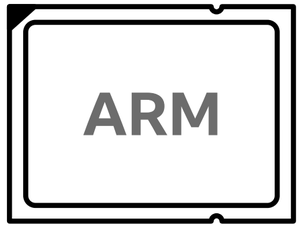
Finally, the way you share your network resources is political, and as such there is no one-size-fit-all policy addressing all use cases.

Since we did not find any off-the-shelf solution ma
network usage of Mesos container
on Linux standard tool
informatio

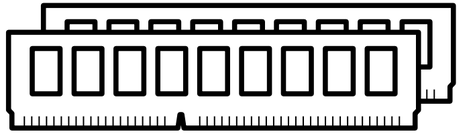


ee: https://medium.com/criteo-engineering/demystification-of-tc-de3dfe4067c2

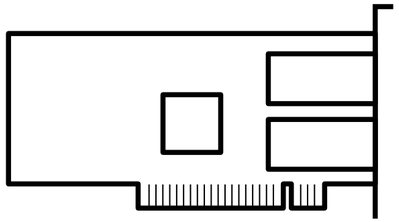
Conclusions



ARM Servers Work Well



Memory Bandwidth now the Bottleneck



Realising NIC Performance is Challenging

BBC

Thank you!

Jonathan Heathcote

BBC Research & Development

jonathan.heathcote@bbc.co.uk

BBC