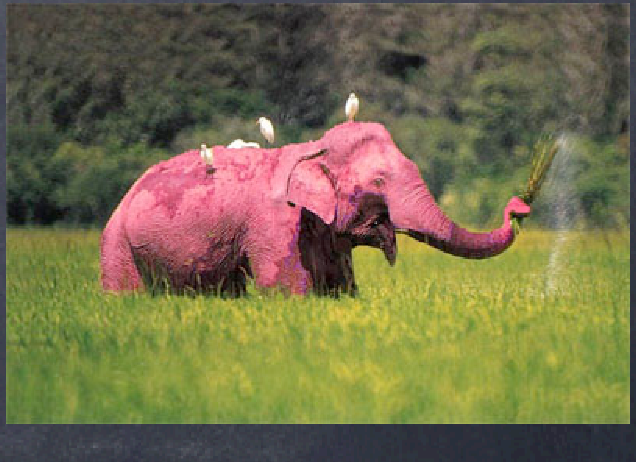
Making Byzantine Fault Tolerant Systems Tolerate Byzantine Failures

Allen Clement, Mirco Marchetti, Edmund Wong
Lorenzo Alvisi, Mike Dahlin



BFT Systems

- PBFT [OSDI 98]
- HQ [OSDI 06]
- Zyzzyva [SOSP 07]
- HT BFT [DSN 04]
- QU [SOSP 05]
- BFT Under Attack [NSDI 08]
- Commit Barrier Scheduling [SOSP 07]
- Low Overhead BFT [SOSP 07]
- Attested Append Only Memory [SOSP 07]
- Beyond 1/3 Faulty in BFT [SOSP 07]
- BASE [OSDI 02]
- SafeStore [USENIX 07]
- Separating Agreement from Execution [SOSP 03]
- SUNDR [OSDI 04]
- ...

System Throughput

	Best Case
PBFT	62k
Q/U	24k
HQ	15k
Zyzzyva	65k

ops/sec

System Throughput

	Best Case	Faulty Client	Client Flood	Faulty Primary	Faulty Replica
PBFT	62k	0	crash	1K	250
Q/U	24k	0	crash	NA	19k
HQ	15k	NA	4.5k	NA	crash
Zyzzyva	65k	0	crash	crash	0

ops/sec

System Throughput

	Best Case	Faulty Client	Client Flood	Faulty Primary	Faulty Replica
PBFT	62k	0	crash	1K	250
Q/U	24k	0	crash	NA	19k
HQ	15k	NA	4.5k	NA	crash
Zyzzyva	65k	0	crash	crash	0
Aardvark	39k	39k	7.8k	37k	11k

ops/sec

Outline

- Robust BFT: The case for a new goal
- Aardvark: Designing for RBFT
- Evaluation: RBFT in action

Paved with good intentions

- No BFT protocol should rely on synchrony for safety
- FLP: No consensus protocol can be both safe and live in an asynchronous system!
  - All one can guarantee is eventual progress
- "Handle normal and worst case separately as a rule, because the requirements for the two are quite different:
  the normal case must be fast;
  the worst case must make some progress"
  -- Butler Lampson, "Hints for Computer System Design"

Recasting the problem

- Maximize performance when
  - the network is synchronous
  - all clients and servers behave correctly
- While remaining
  - safe if at most f servers fails
  - eventually live

Recasting the problem

- Misguided
- Dangerous
- Futile

Recasting the problem

- Misguided
  - it encourages systems that fail to deliver BFT
- Dangerous
- Futile

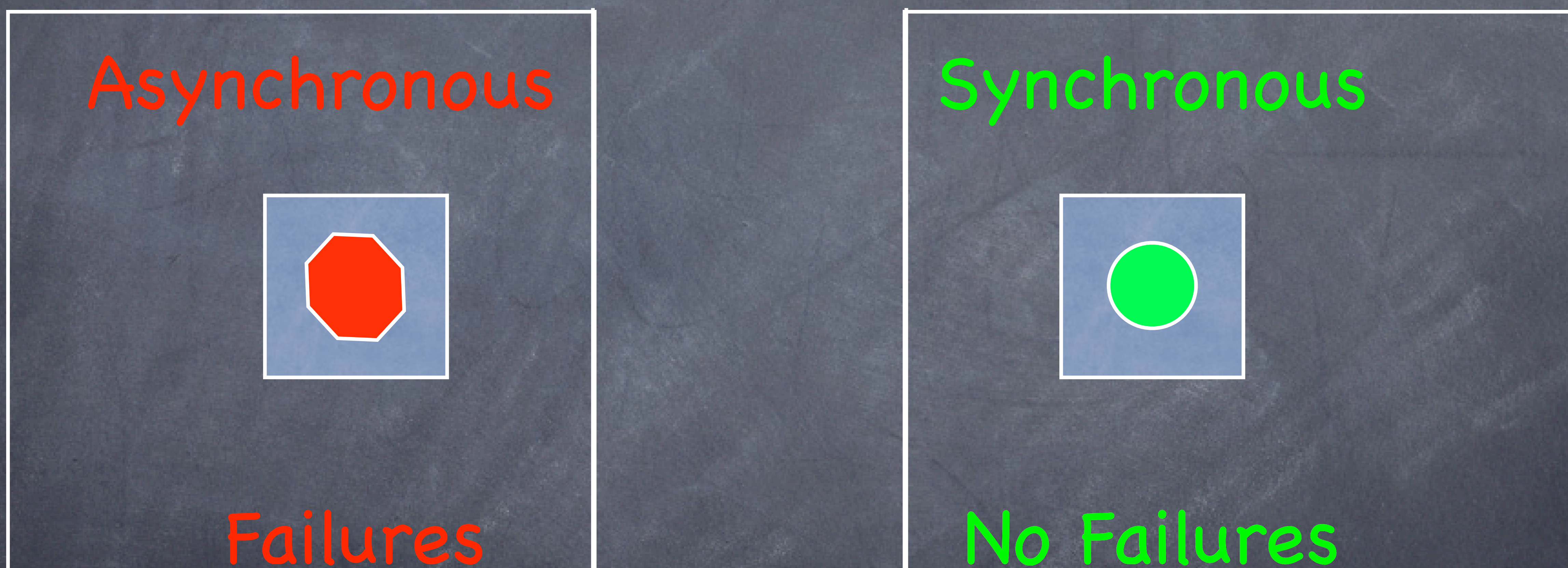
Recasting the problem

- Misguided
  - it encourages systems that fail to deliver BFT
- Dangerous
  - it encourages fragile optimizations
- Futile

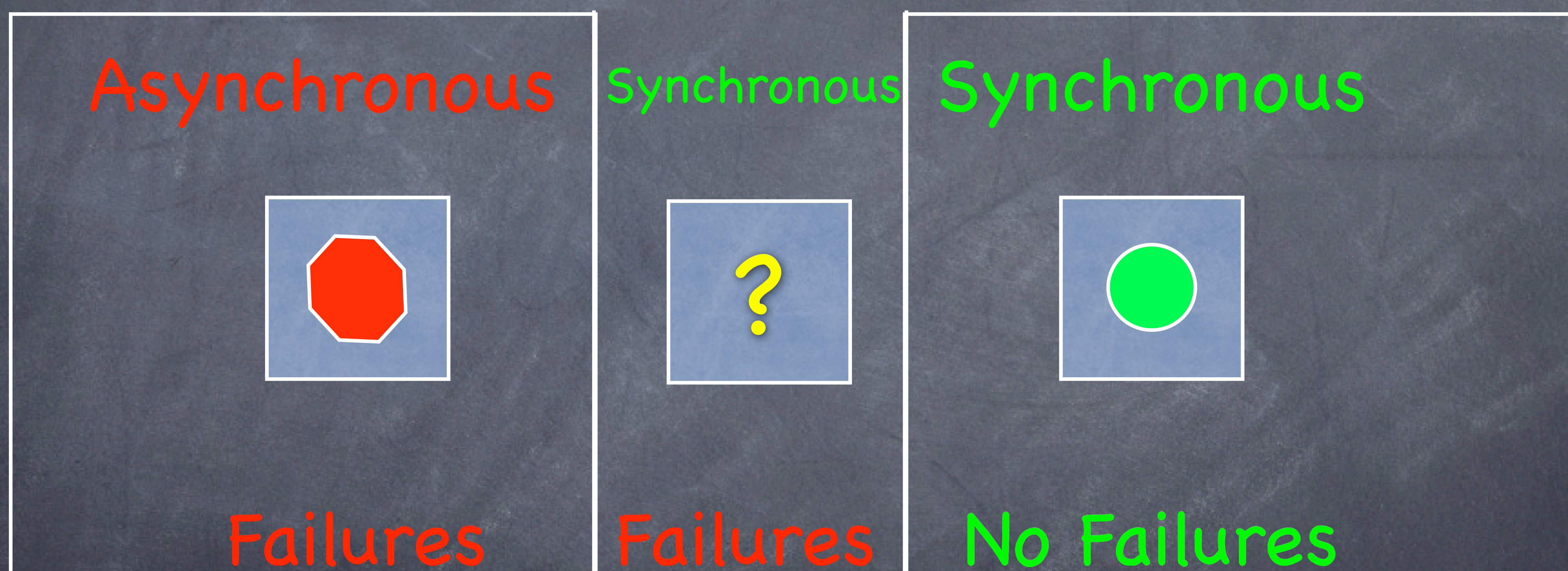
Recasting the problem

- Misguided
  - it encourages systems that fail to deliver BFT
- Dangerous
  - it encourages fragile optimizations
- Futile
  - it yields diminishing return on common case

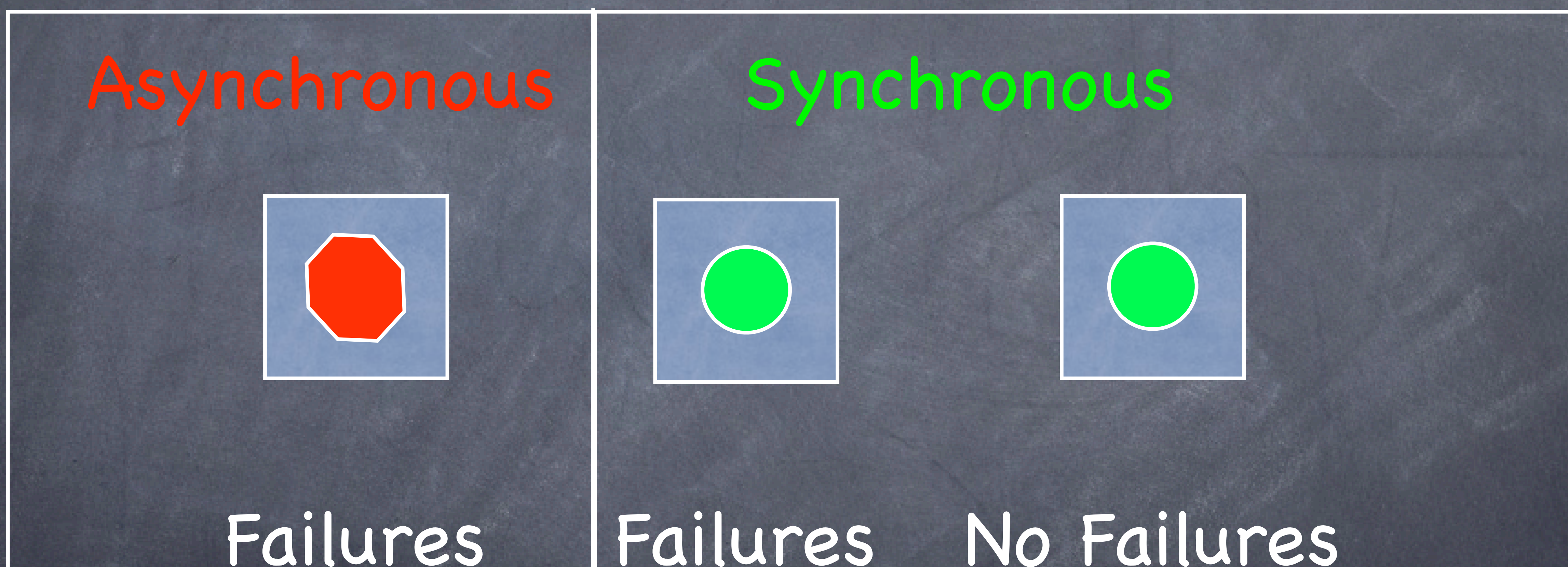
A New Goal



A New Goal


Asynchronous
Synchronous
Synchronous
?
Failures
Failures
No Failures

A New Goal



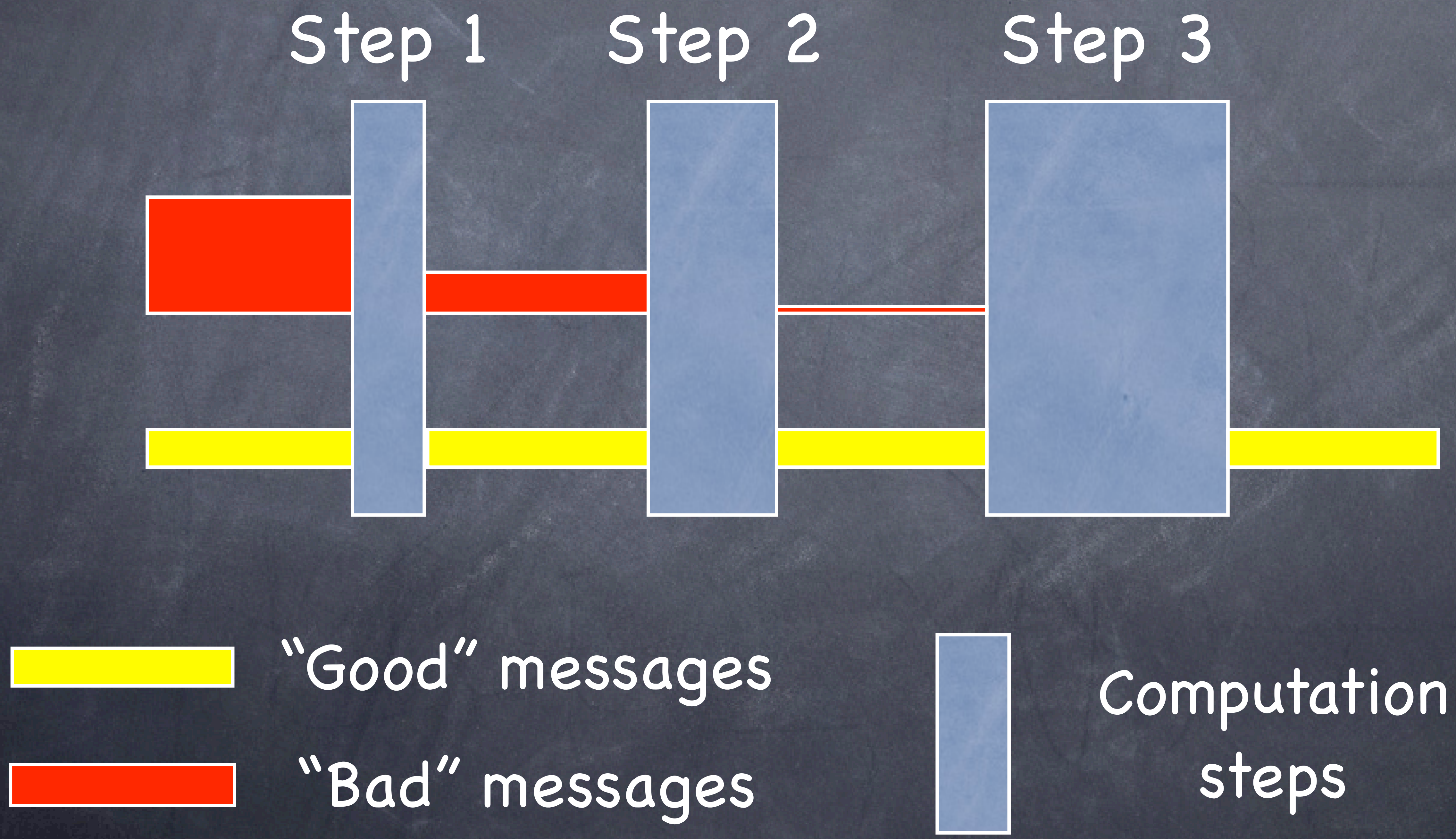
Robust BFT

- Maximize performance when
  - the network is synchronous
  - at most f servers fail
- While remaining
  - safe if at most f servers fail
  - eventually live

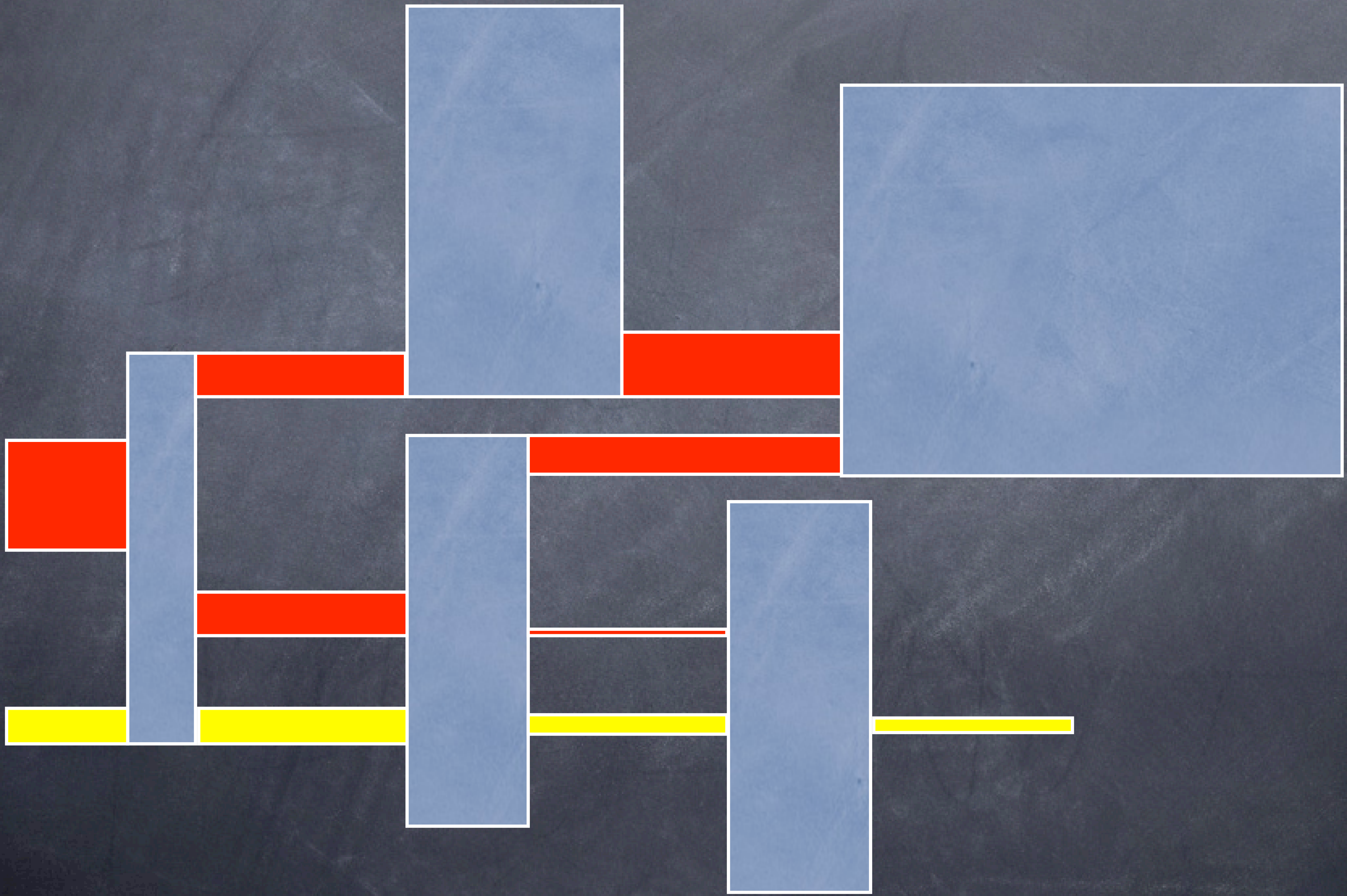
Outline

- Robust BFT: The case for a new goal
- Aardvark: Designing for RBFT
- Evaluation: RBFT in action

Protocol Structure



Fragile Optimizations



Revisiting conventional wisdom

- Signatures are expensive - use MACs
- View changes are to be avoided
- Hardware multicast is a boon

Revisiting conventional wisdom

- Signatures are expensive - use MACs
  - Faulty clients can use MACs to generate ambiguity
  - Aardvark requires clients to sign requests
- View changes are to be avoided
- Hardware multicast is a boon

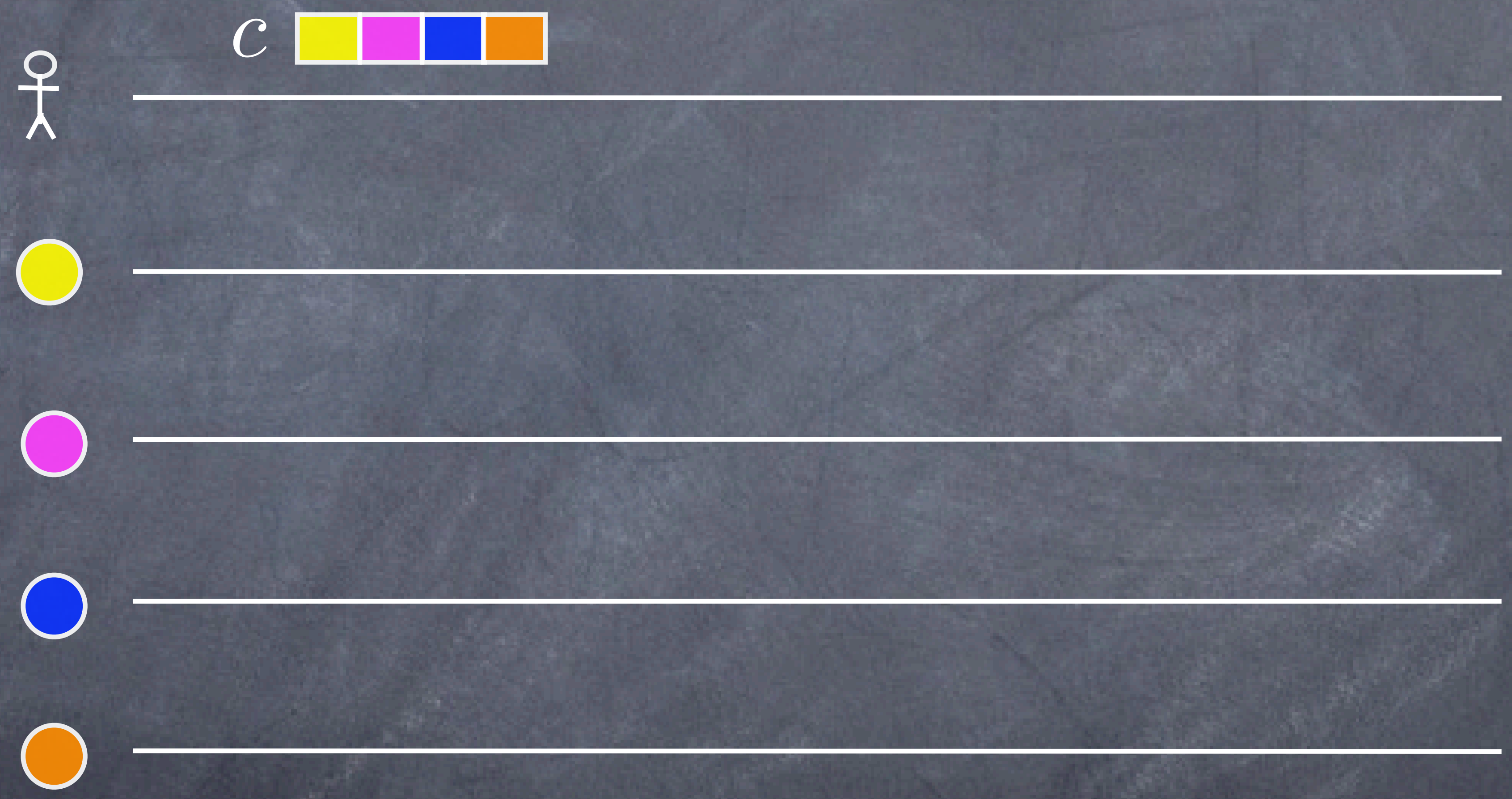
Revisiting conventional wisdom

- Signatures are expensive - use MACs
  - Faulty clients can use MACs to generate ambiguity
  - Aardvark requires clients to sign requests
- View changes are to be avoided
  - Aardvark uses regular view changes to maintain high throughput despite faulty primaries
- Hardware multicast is a boon

Revisiting conventional wisdom

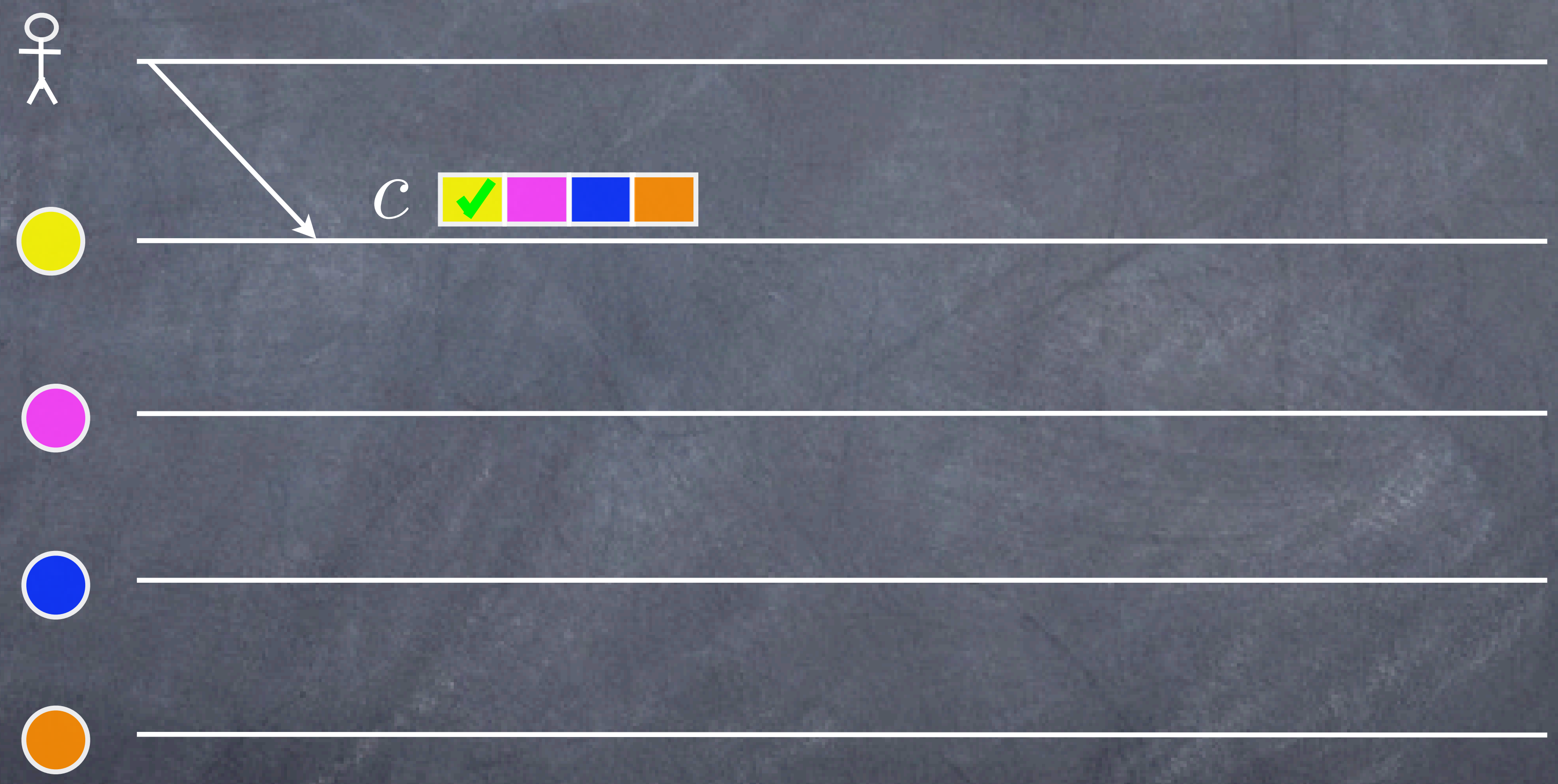
- Signatures are expensive - use MACs
  - Faulty clients can use MACs to generate ambiguity
  - Aardvark requires clients to sign requests
- View changes are to be avoided
  - Aardvark uses regular view changes to maintain high throughput despite faulty primaries
- Hardware multicast is a boon
  - Aardvark uses separate work queues for clients and individual replicas

Big MAC Attack

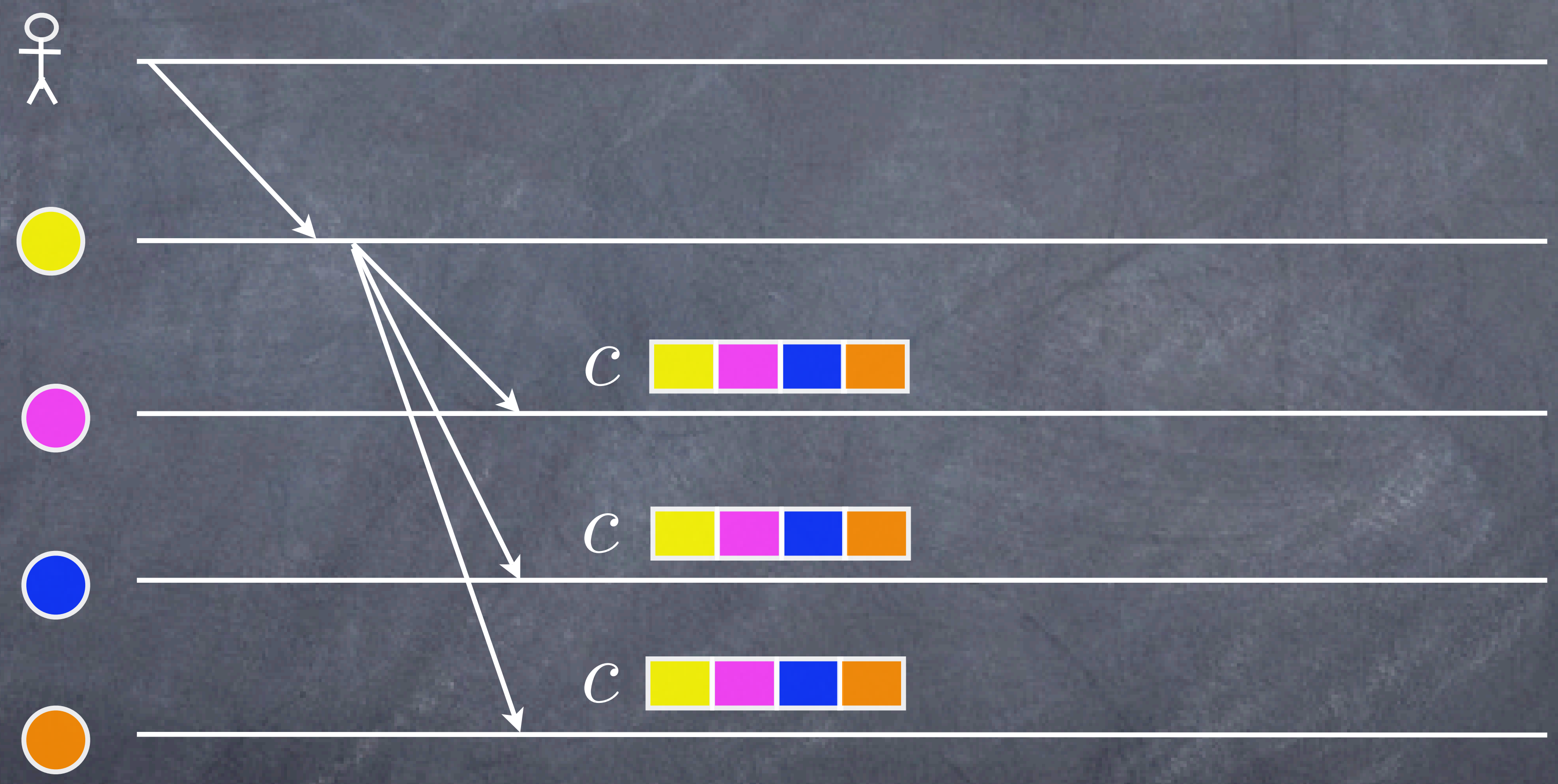


Big MAC Attack

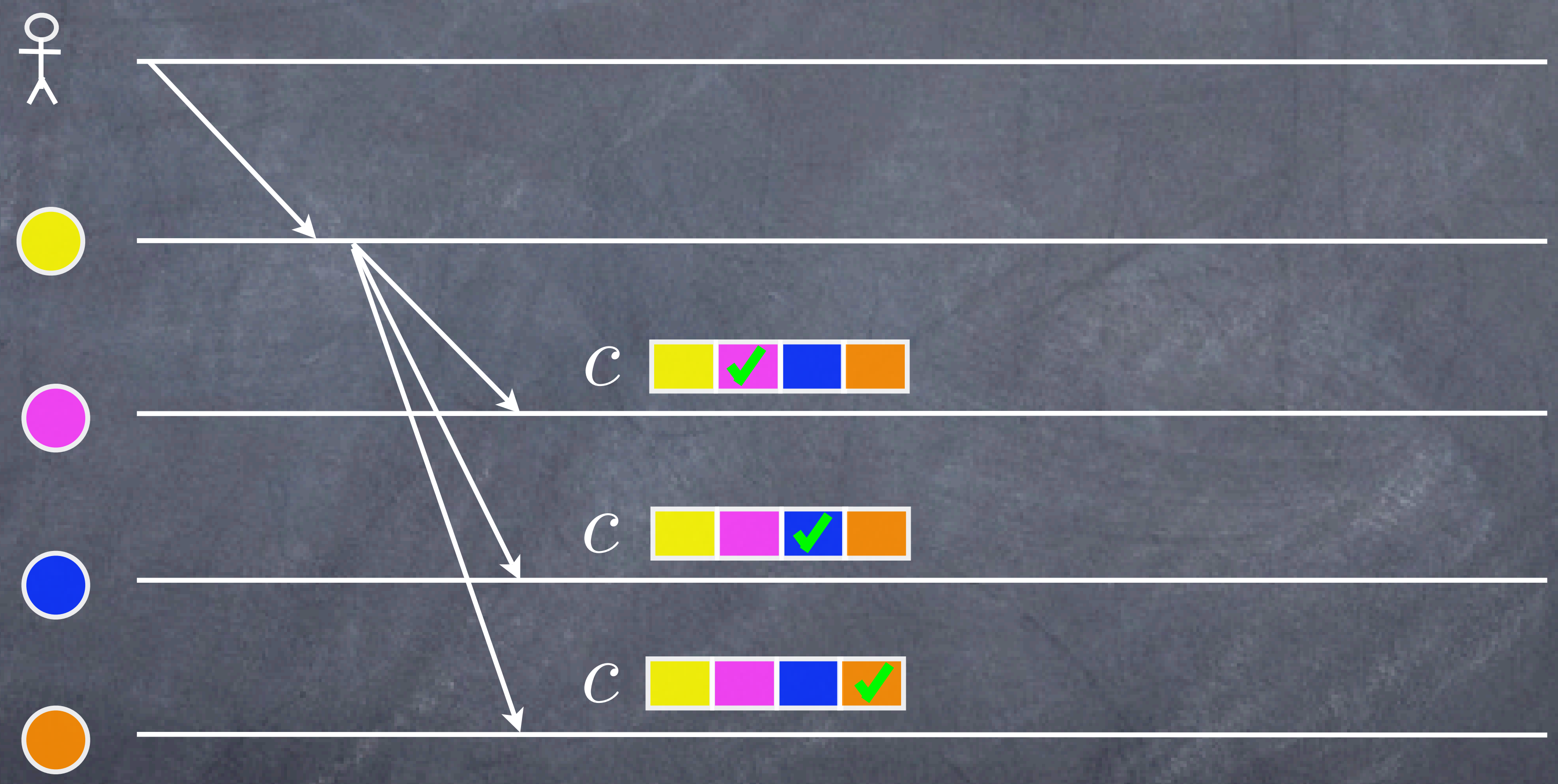
Big MAC Attack



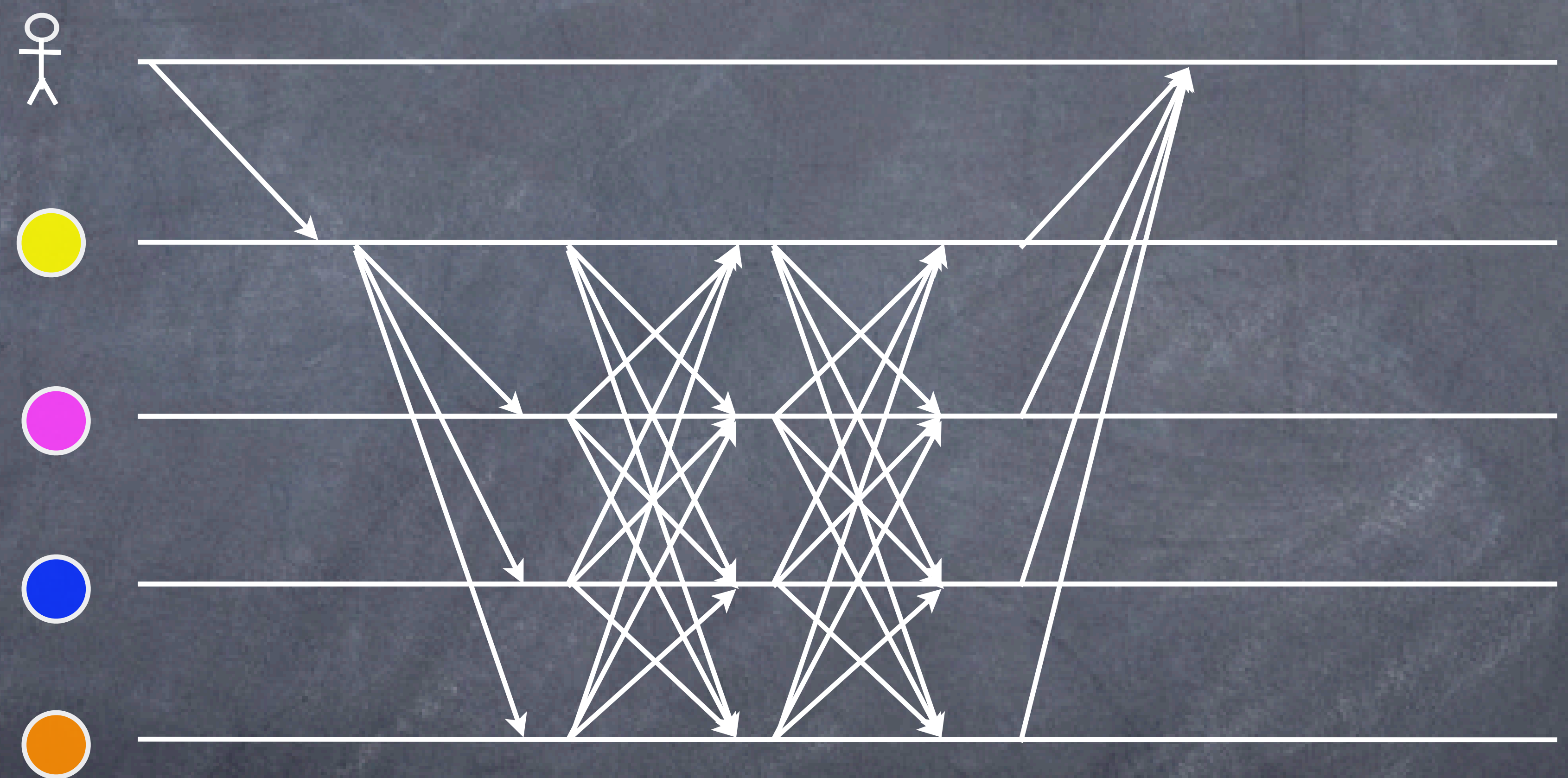
Big MAC Attack



Big MAC Attack

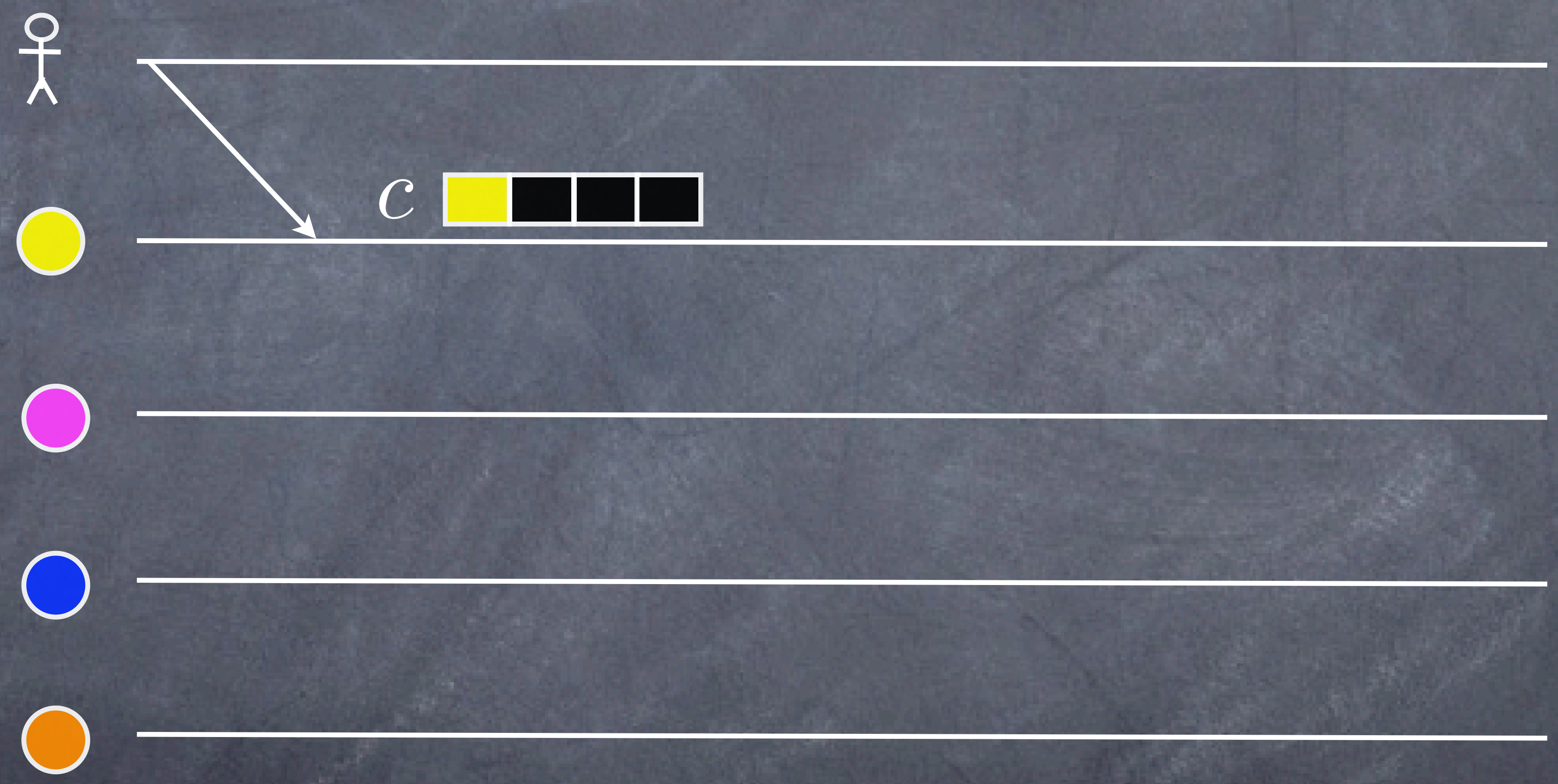


Big MAC Attack

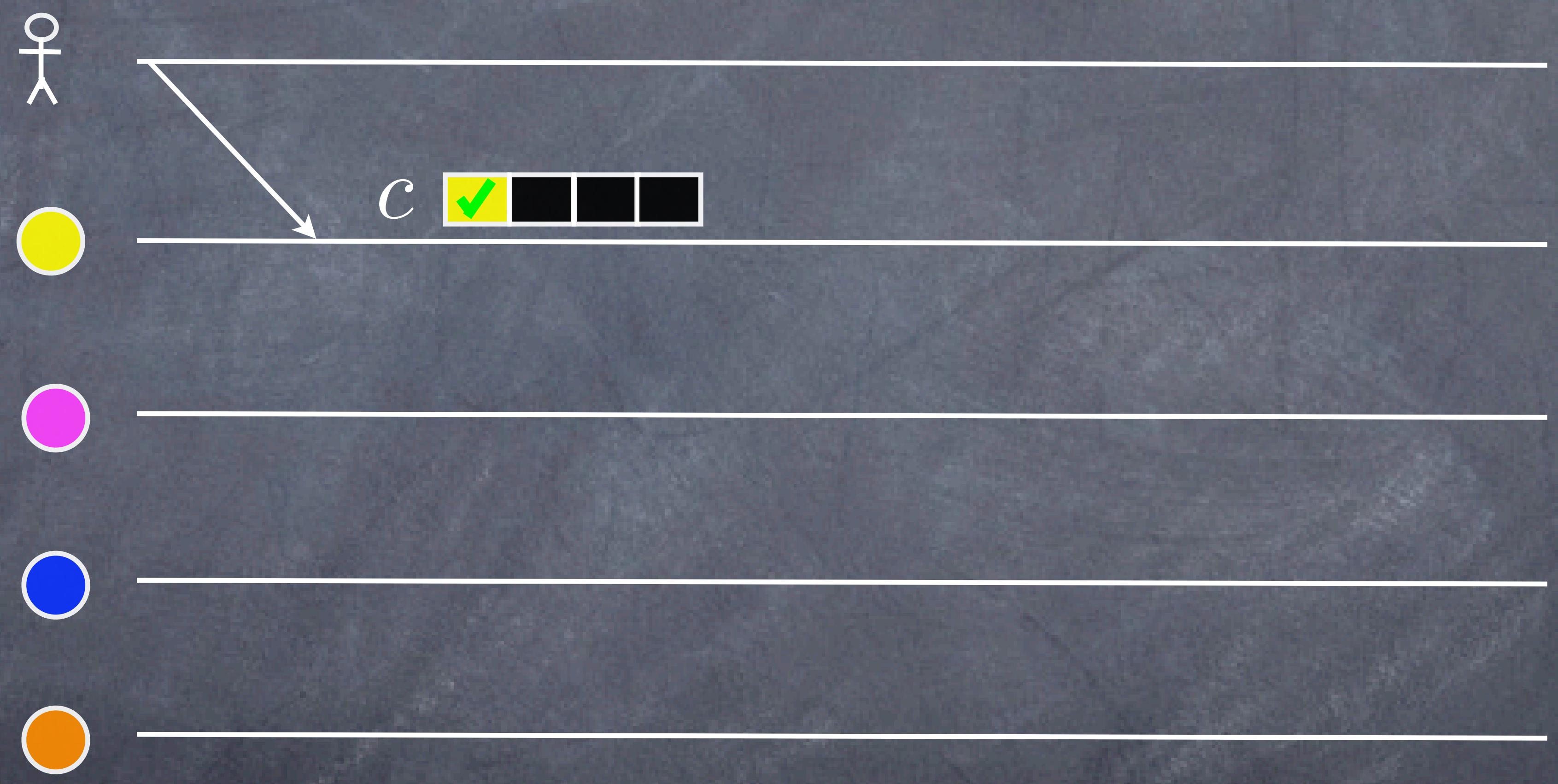


Big MAC Attack

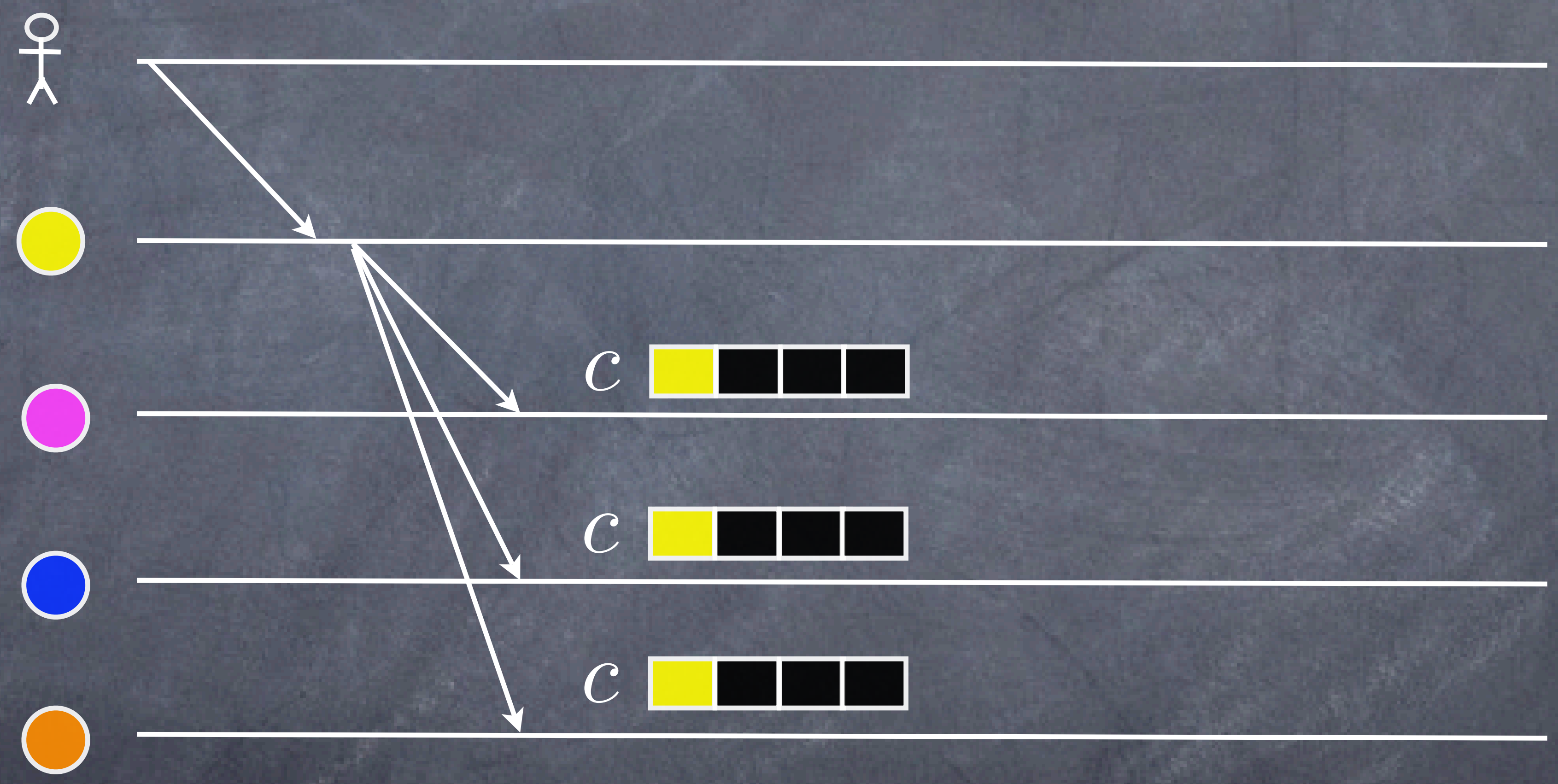
Big MAC Attack



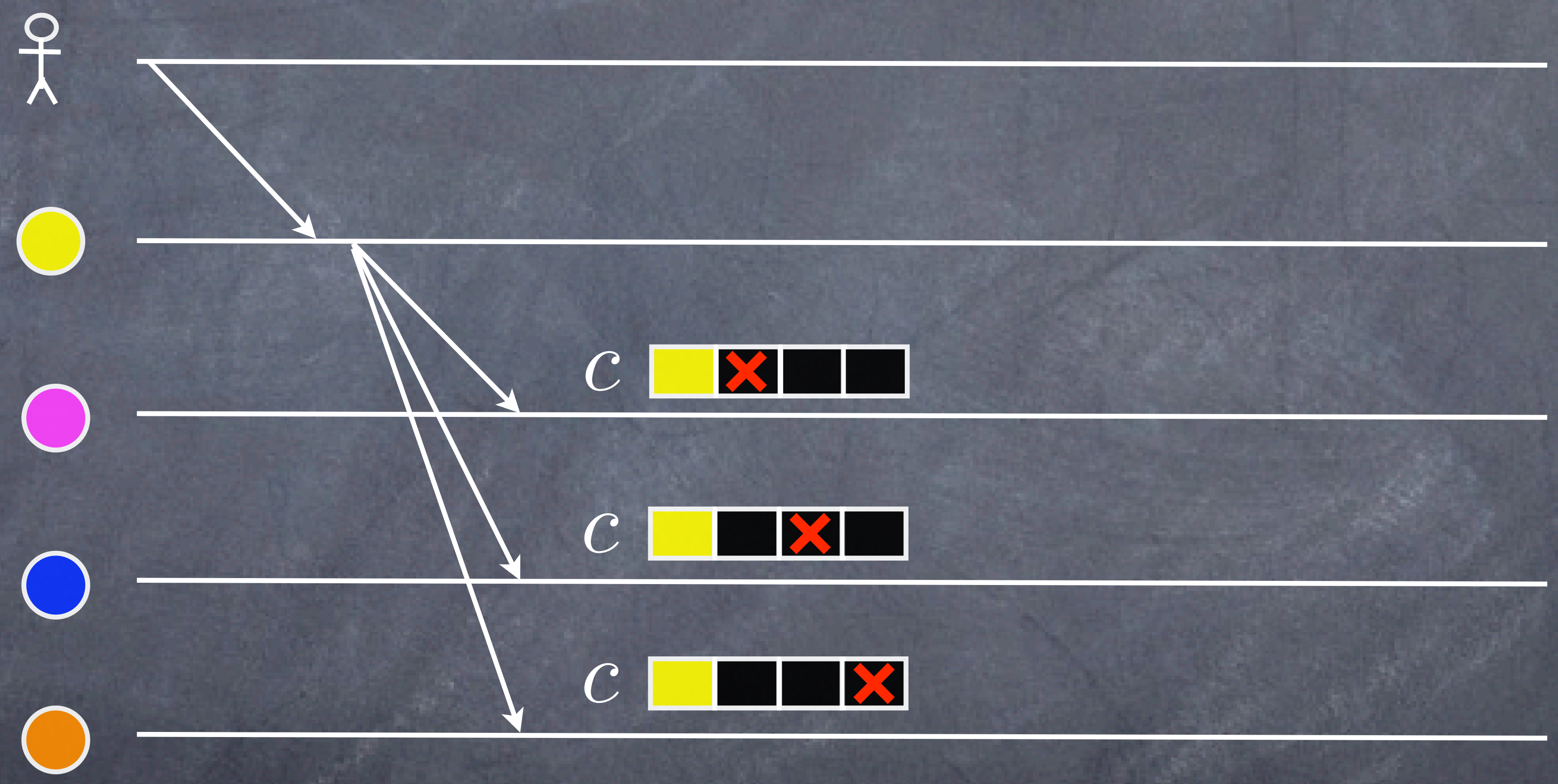
Big MAC Attack



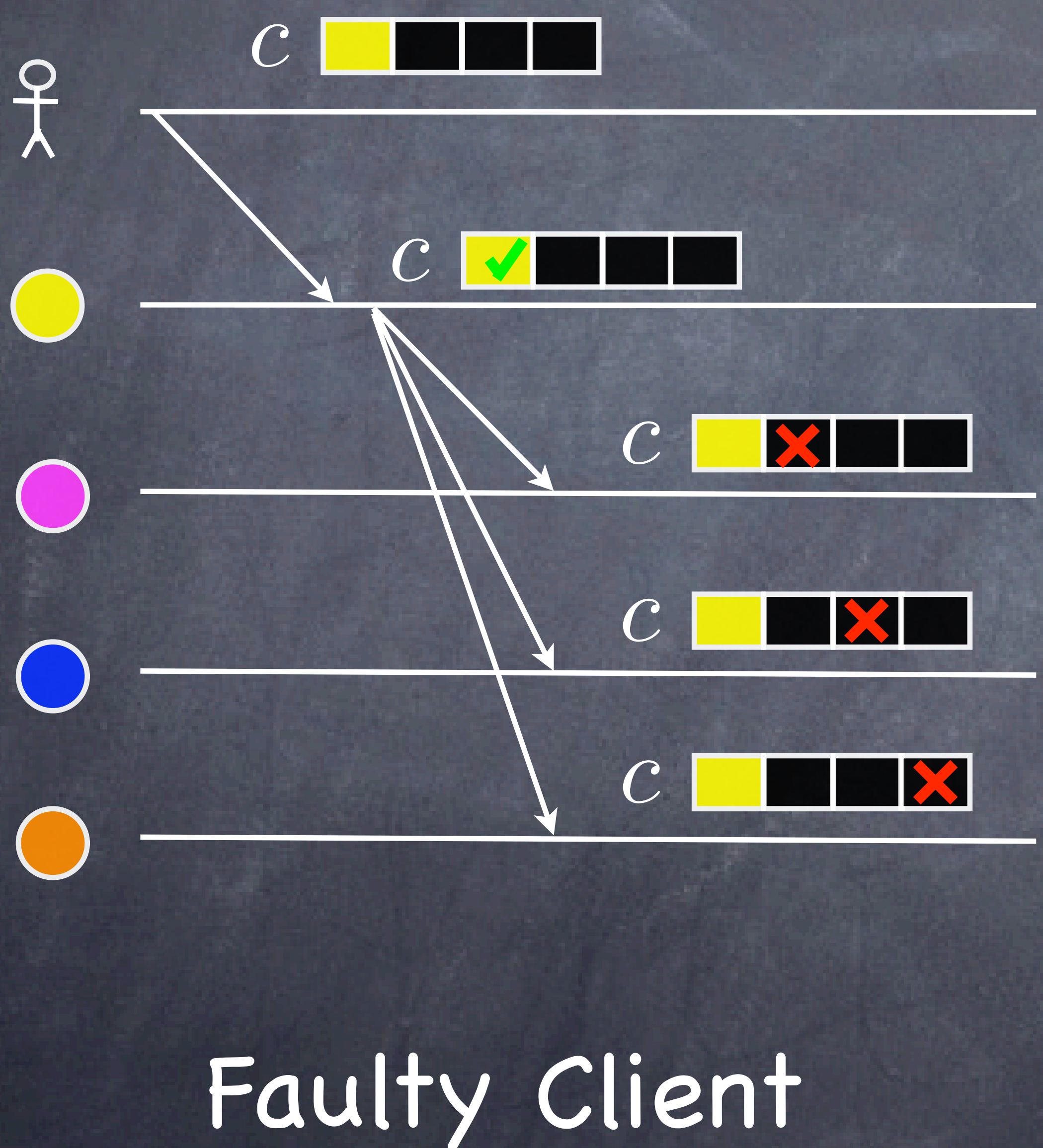
Big MAC Attack

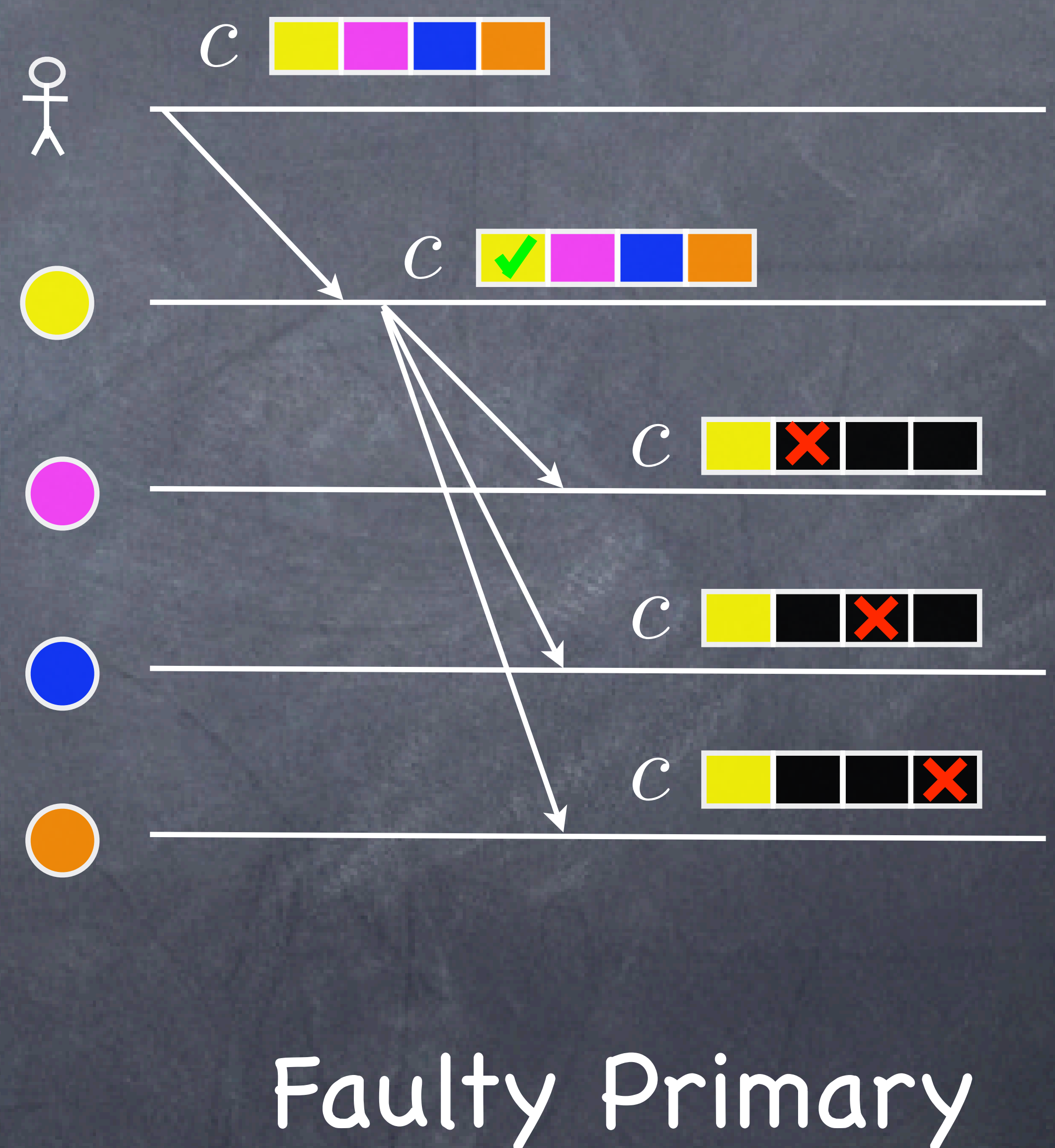


Big MAC Attack



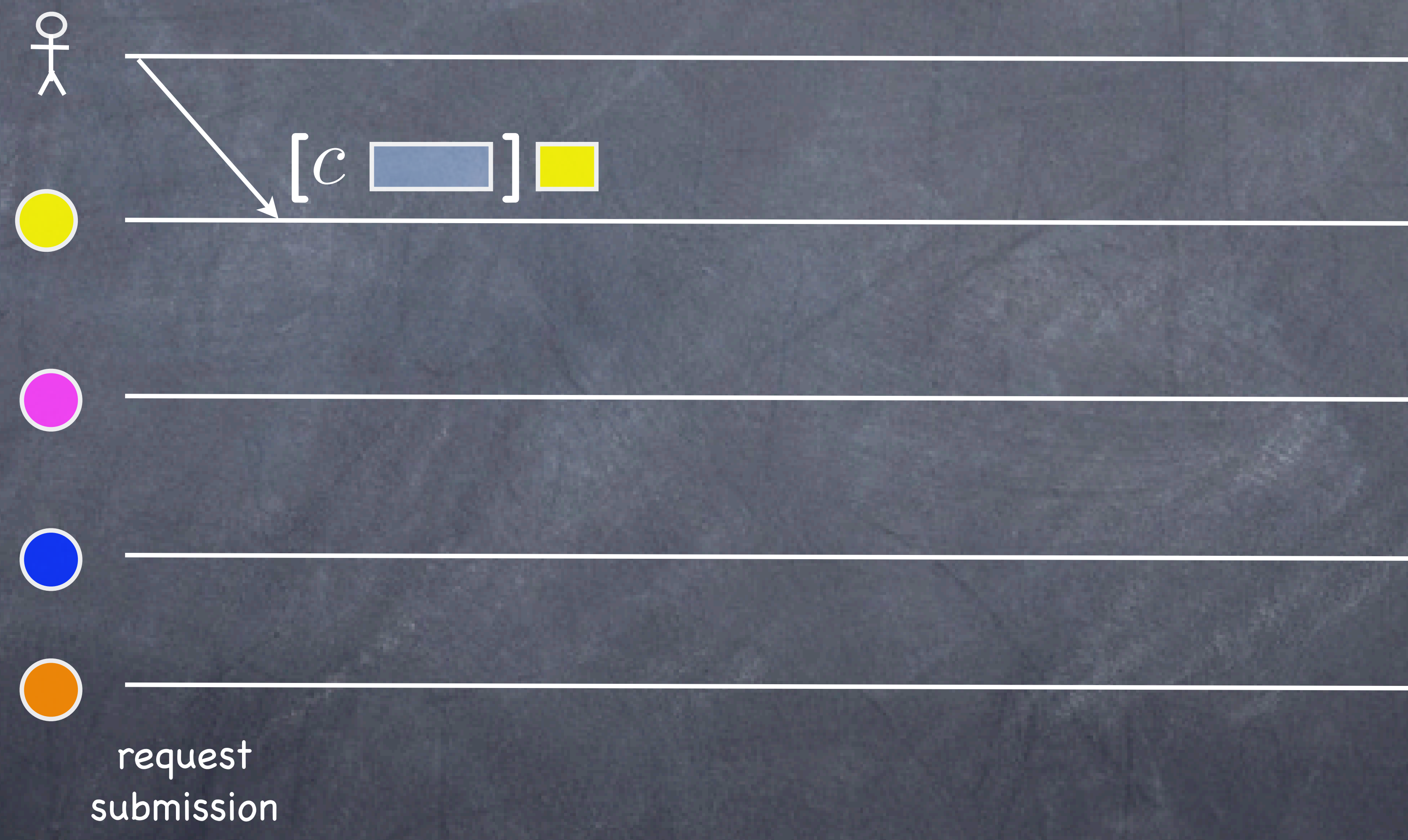
Big MAC Attack



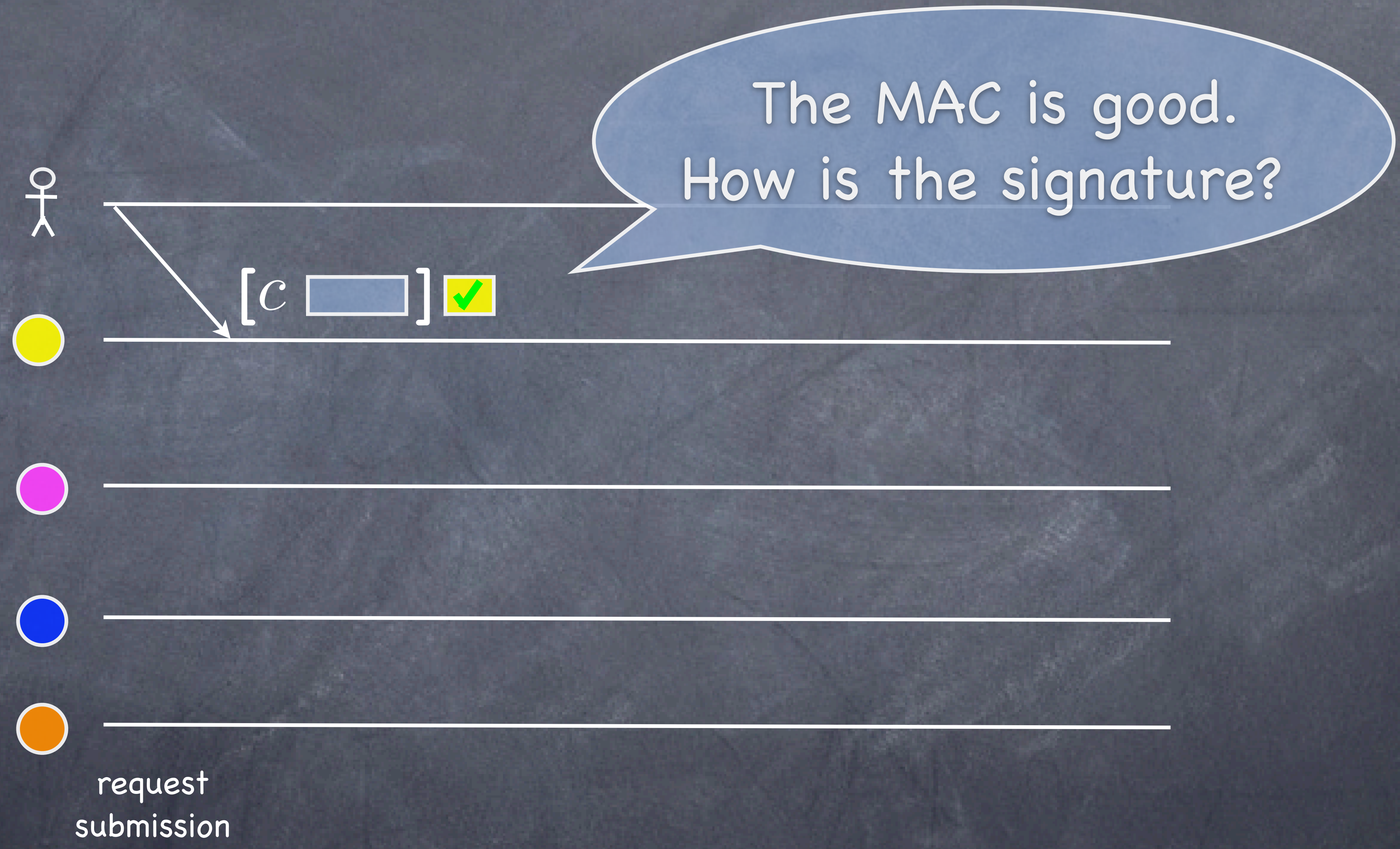


Hybrid MAC/Signatures

Hybrid MAC/Signatures

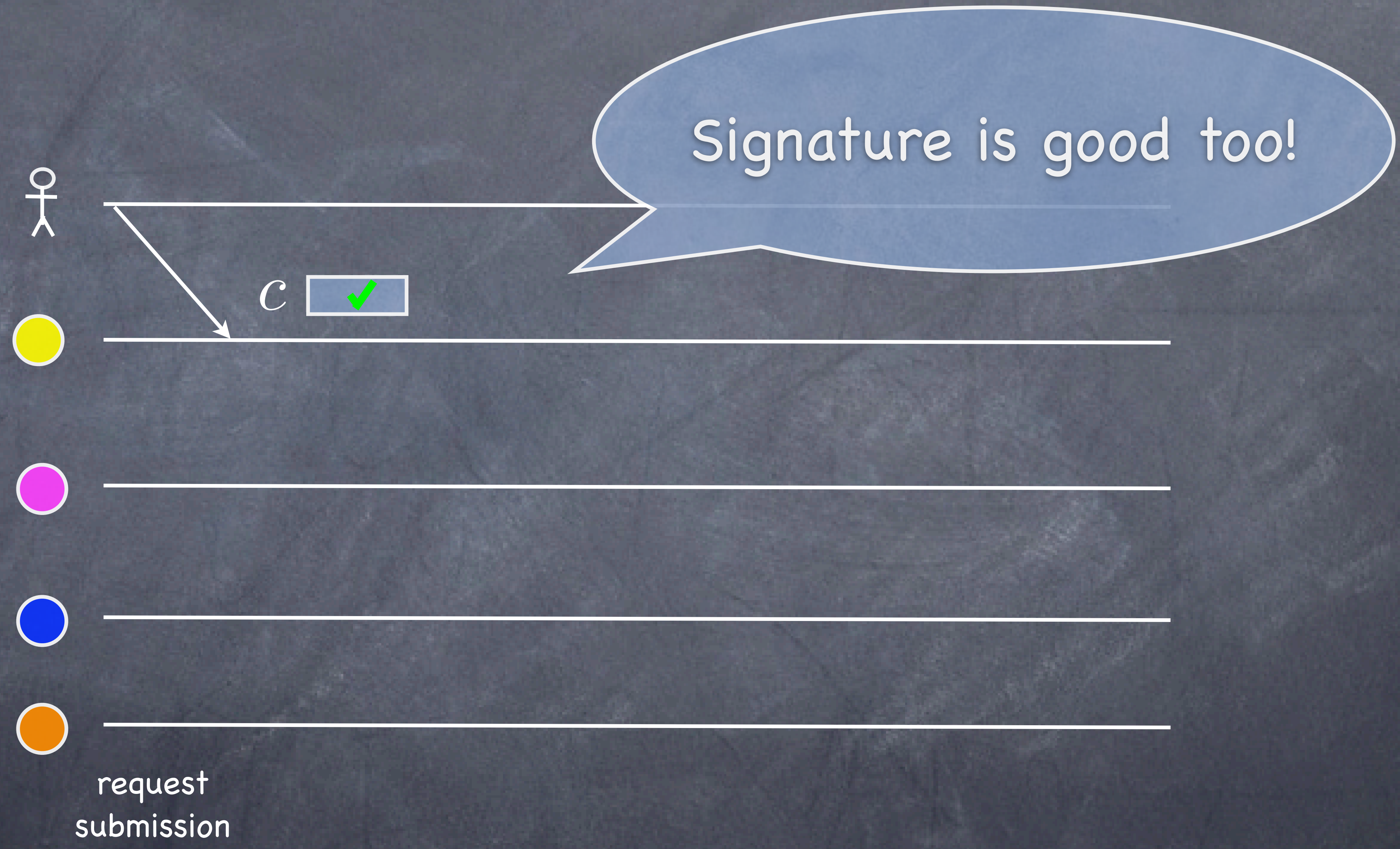


Hybrid MAC/Signatures



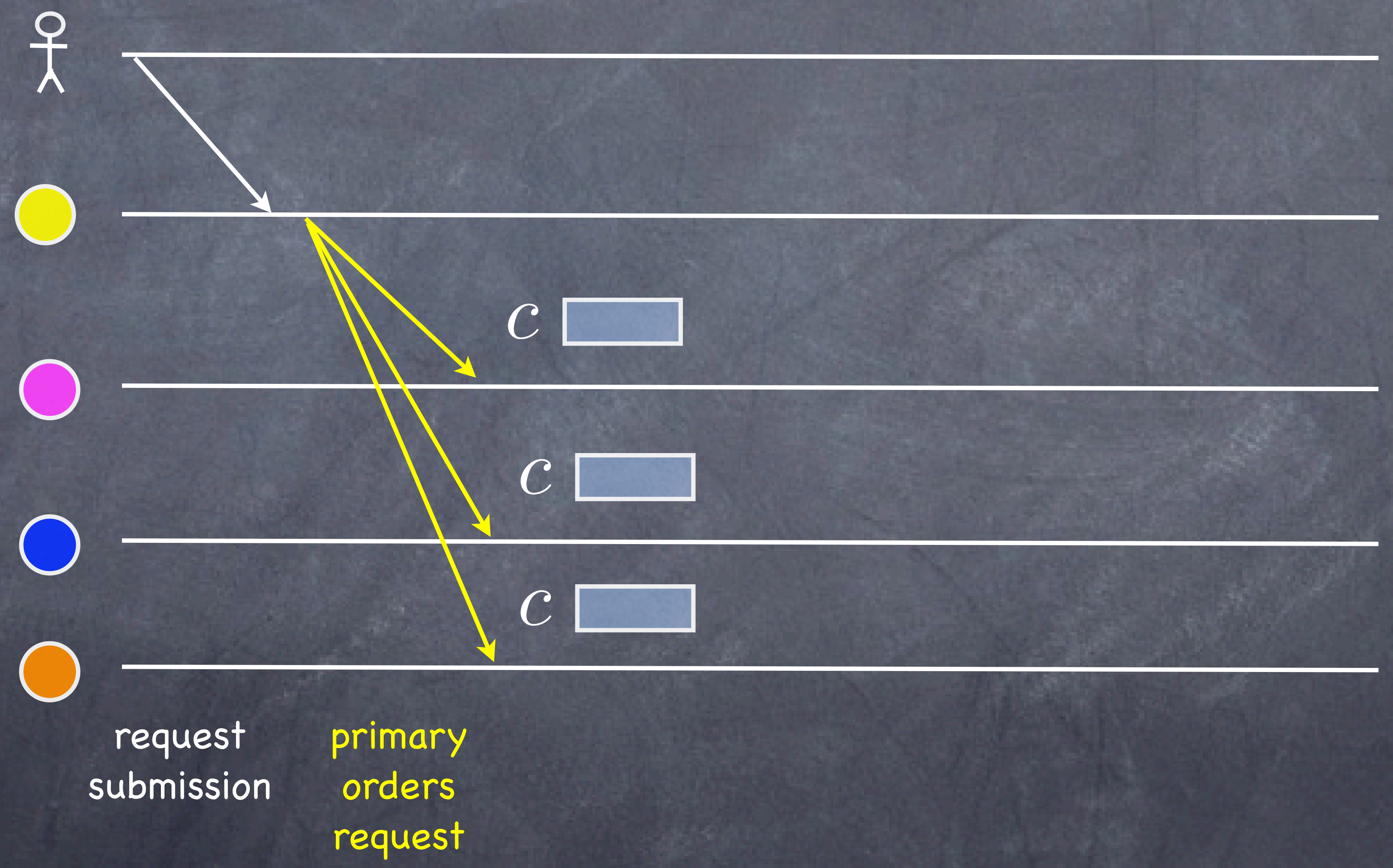
Hybrid MAC/Signatures

Hybrid MAC/Signatures

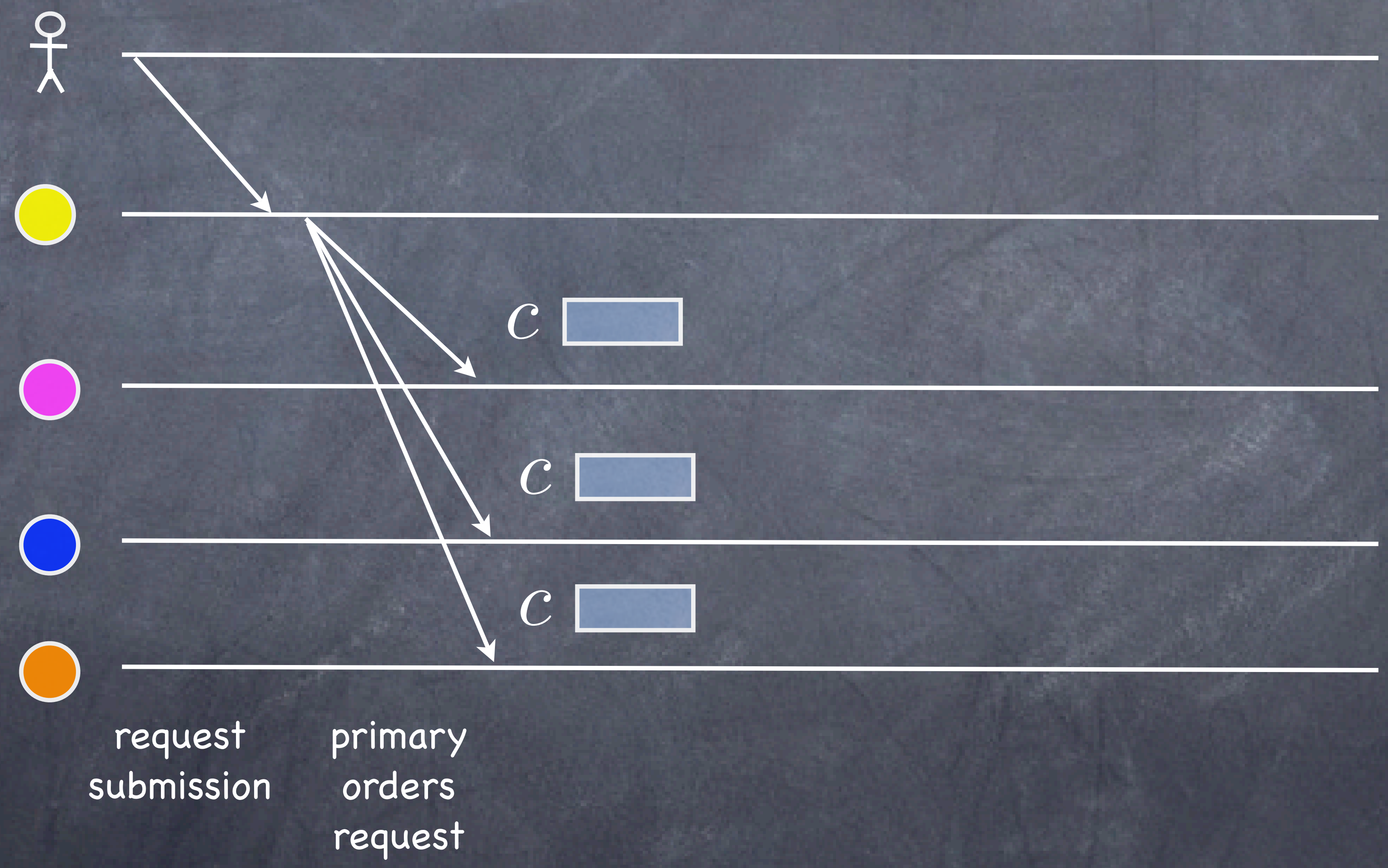


Hybrid MAC/Signatures

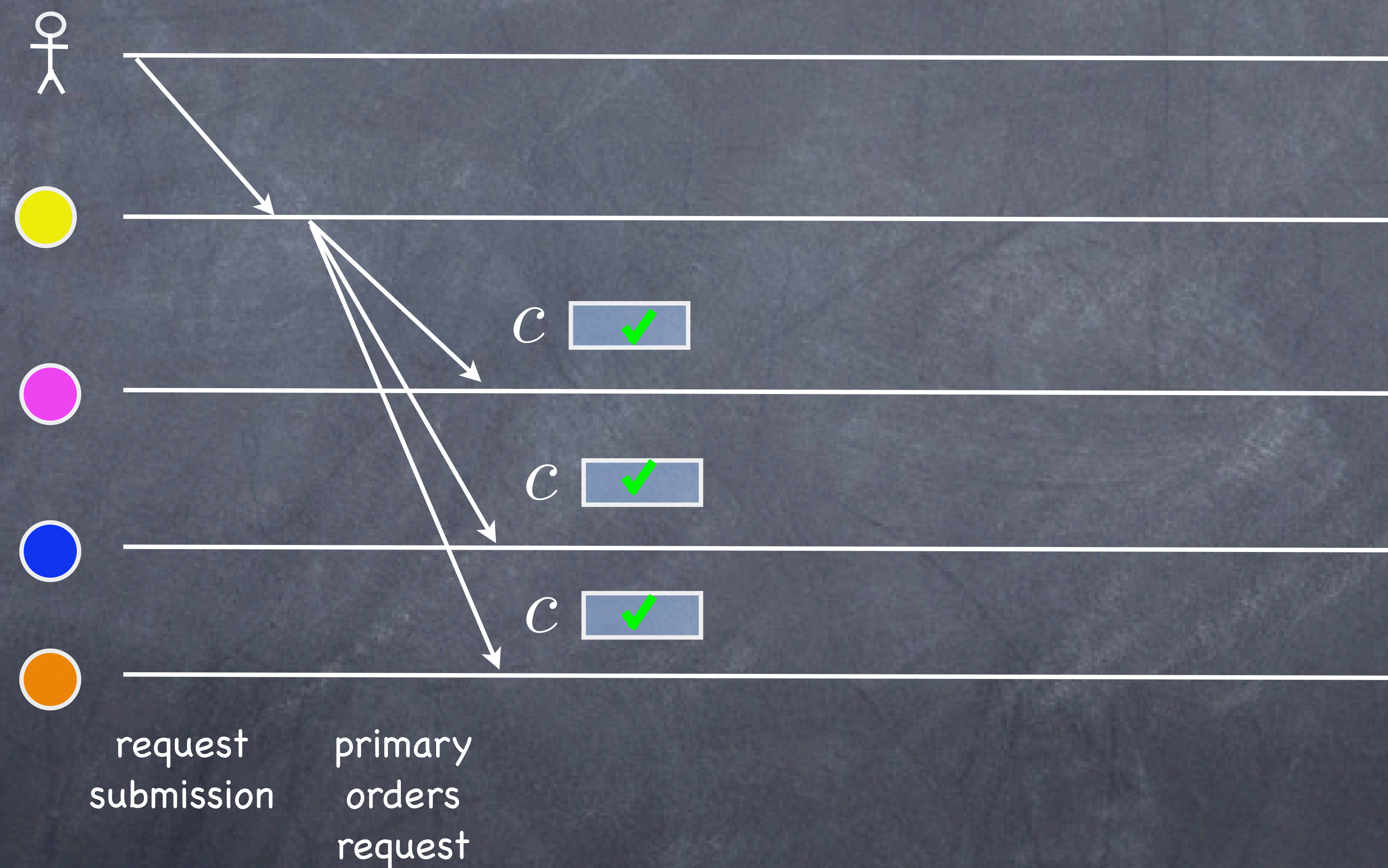
Hybrid MAC/Signatures



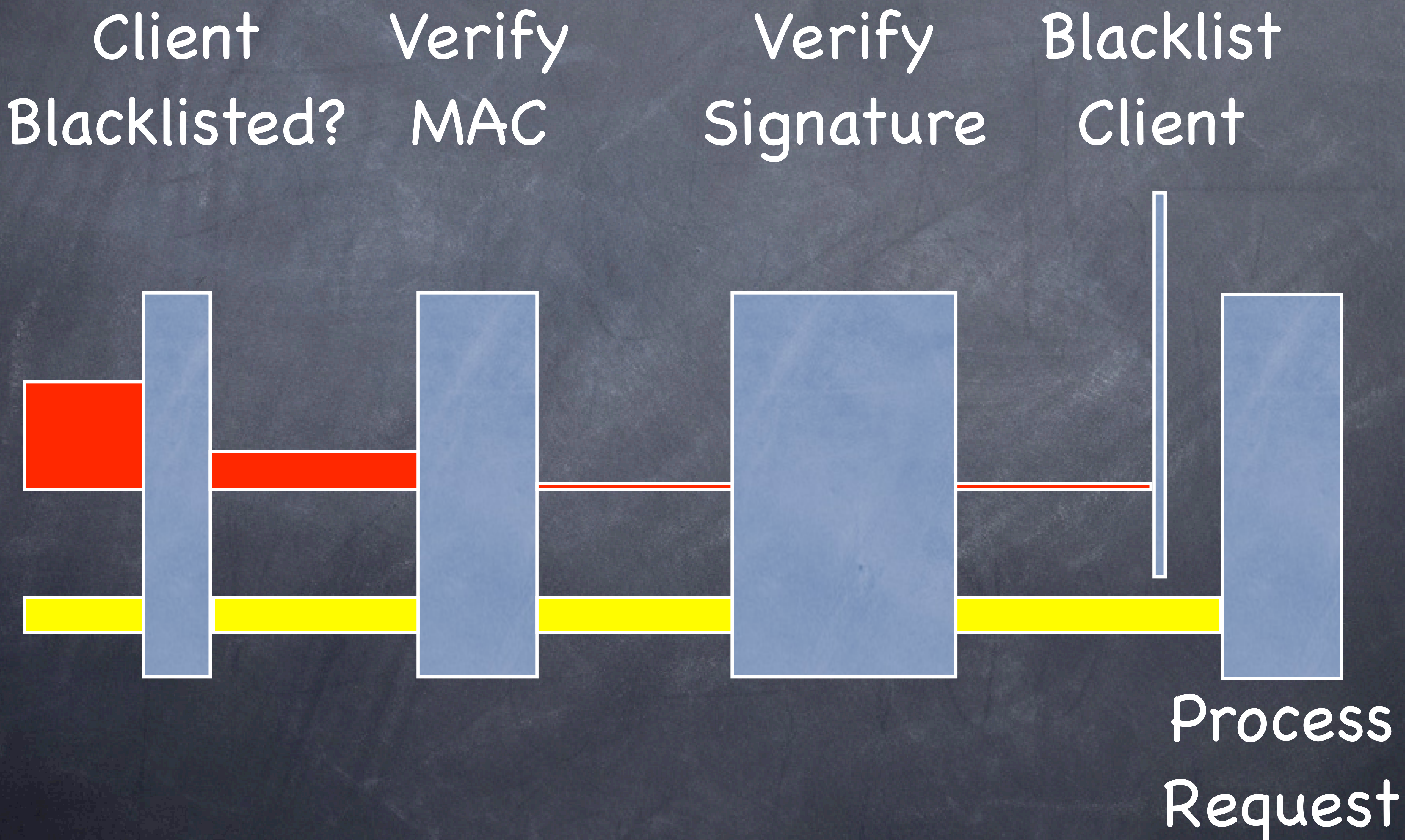
Hybrid MAC/Signatures



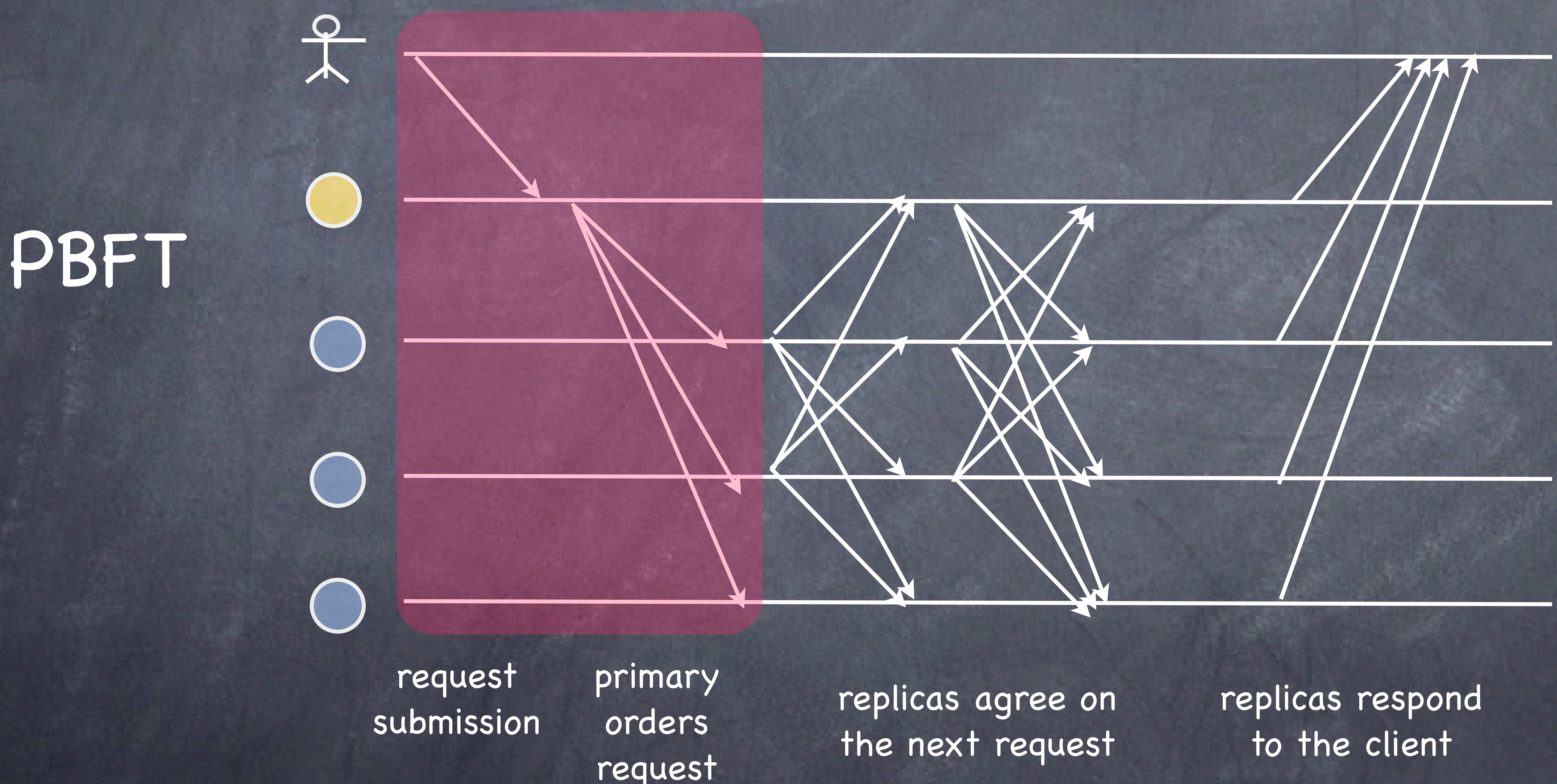
Hybrid MAC/Signatures



Signed Request Filtering

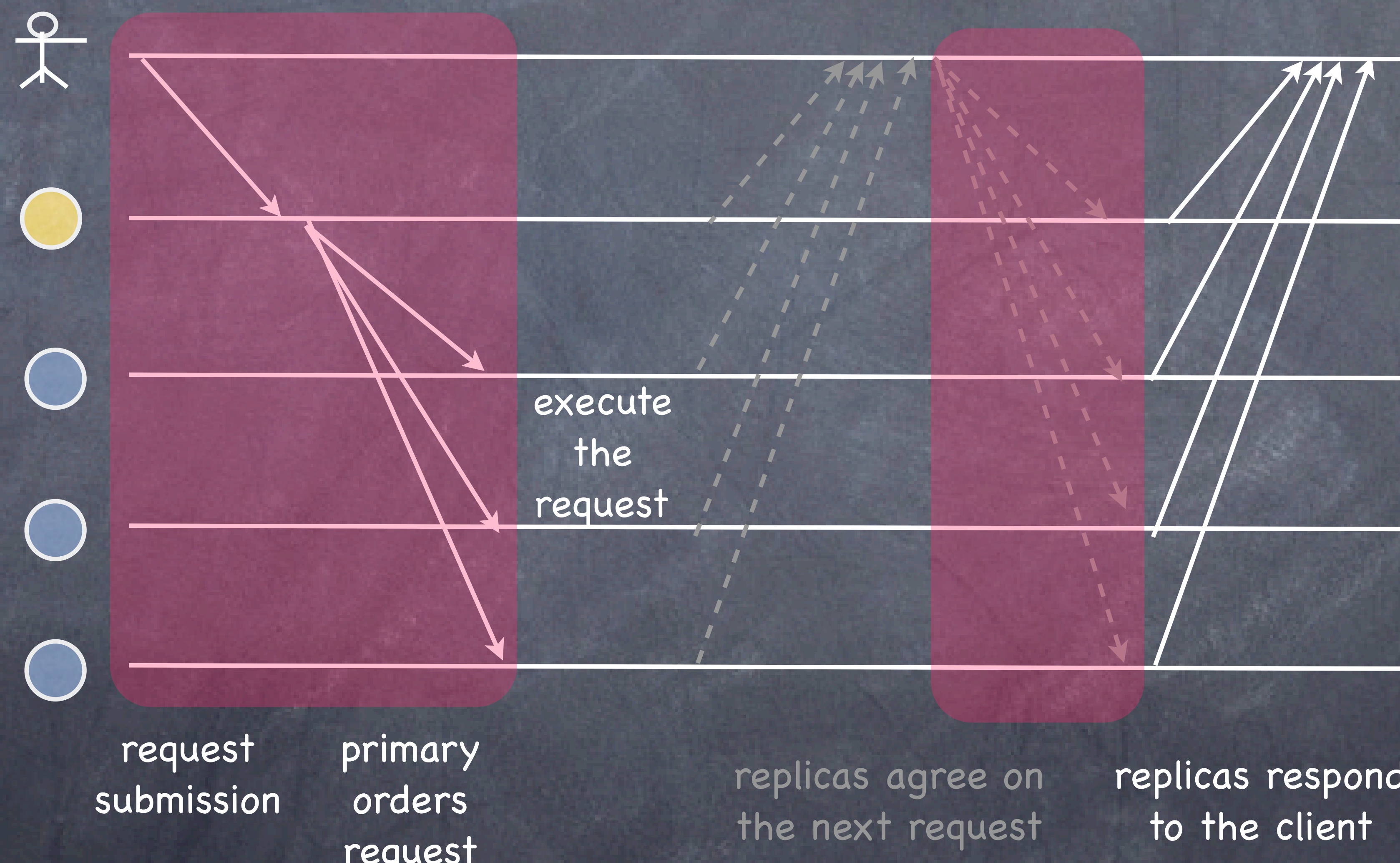


Big MAC Attack



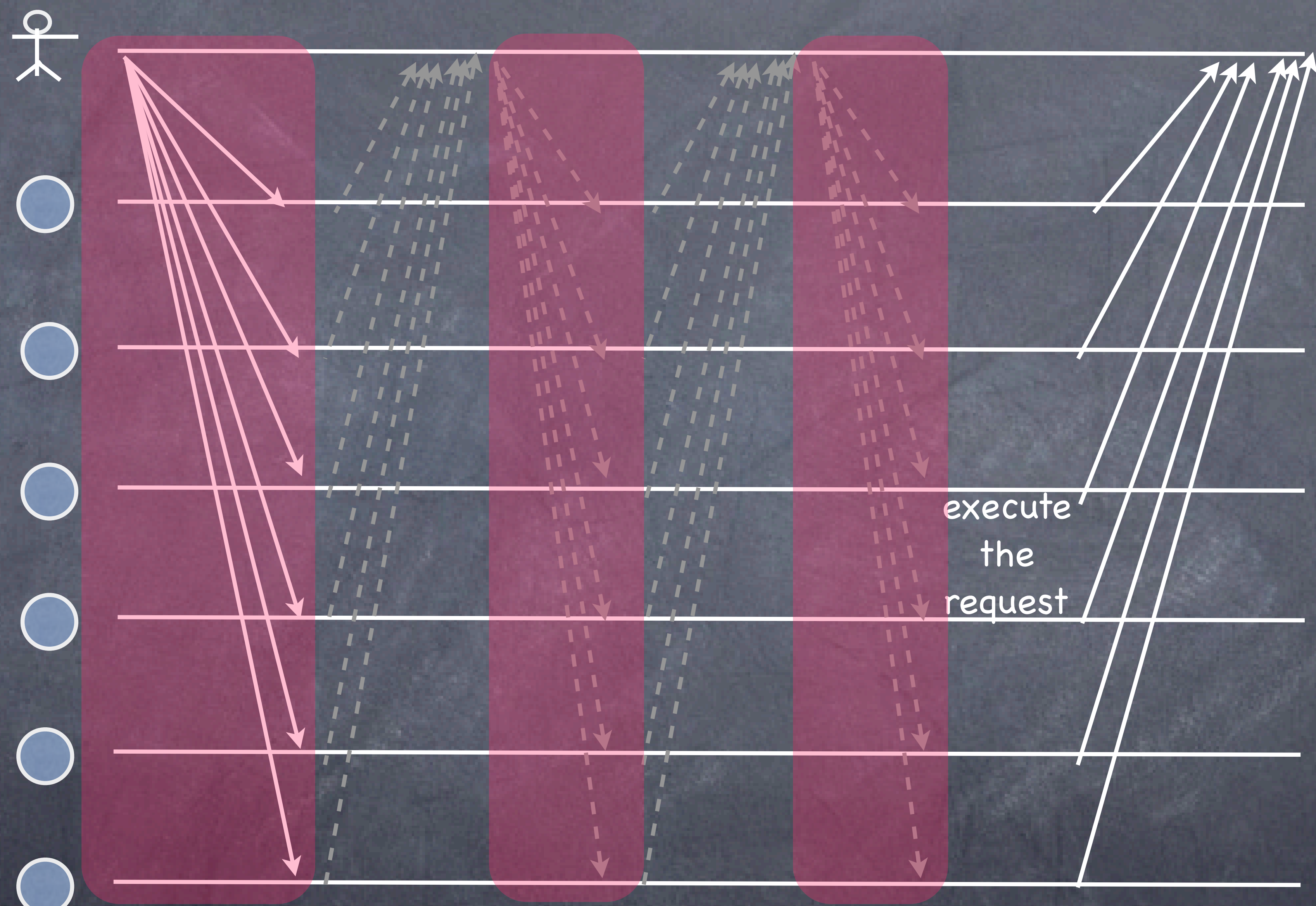
Big MAC Attack

Zyzzyva

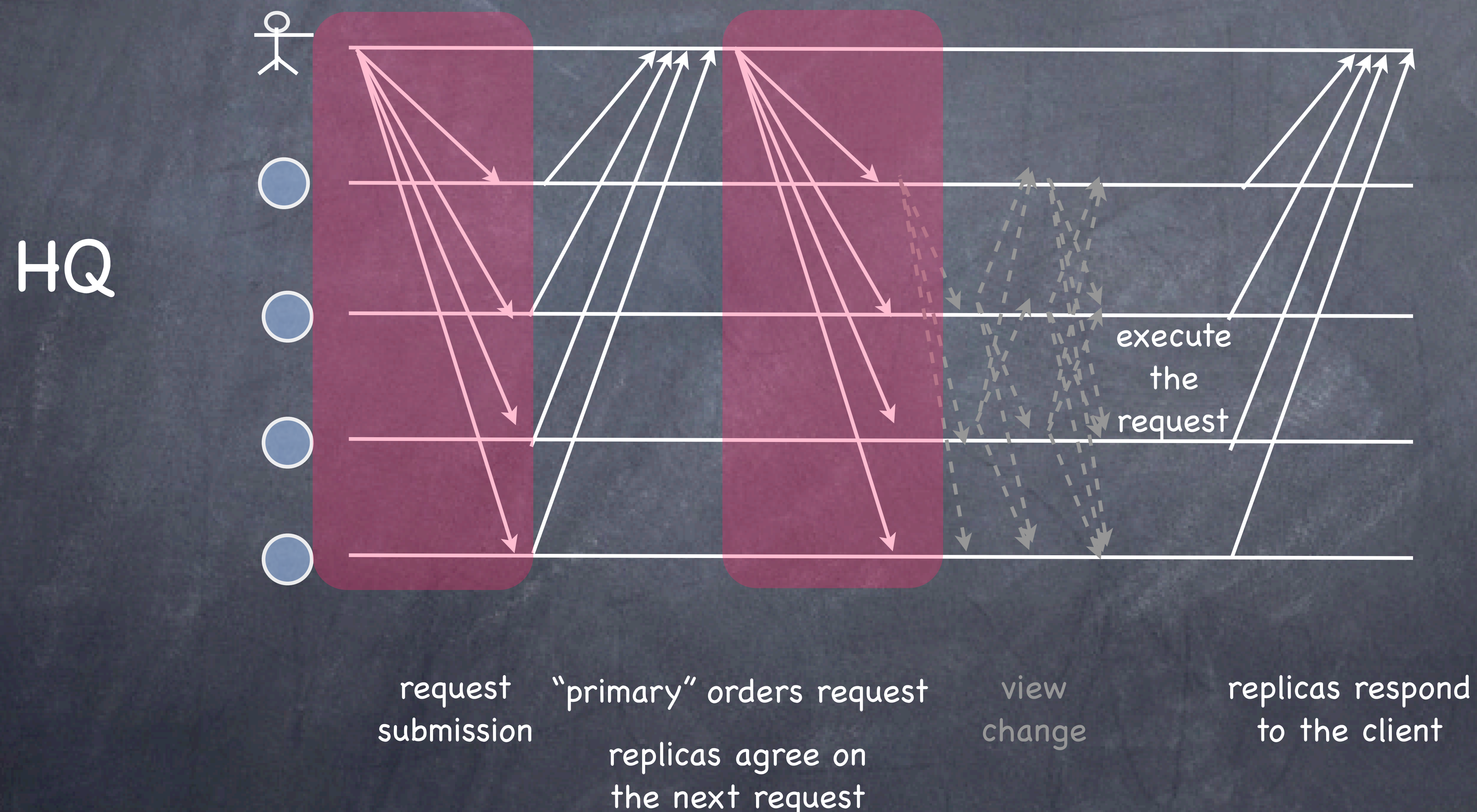


Big MAC Attack

Q/U



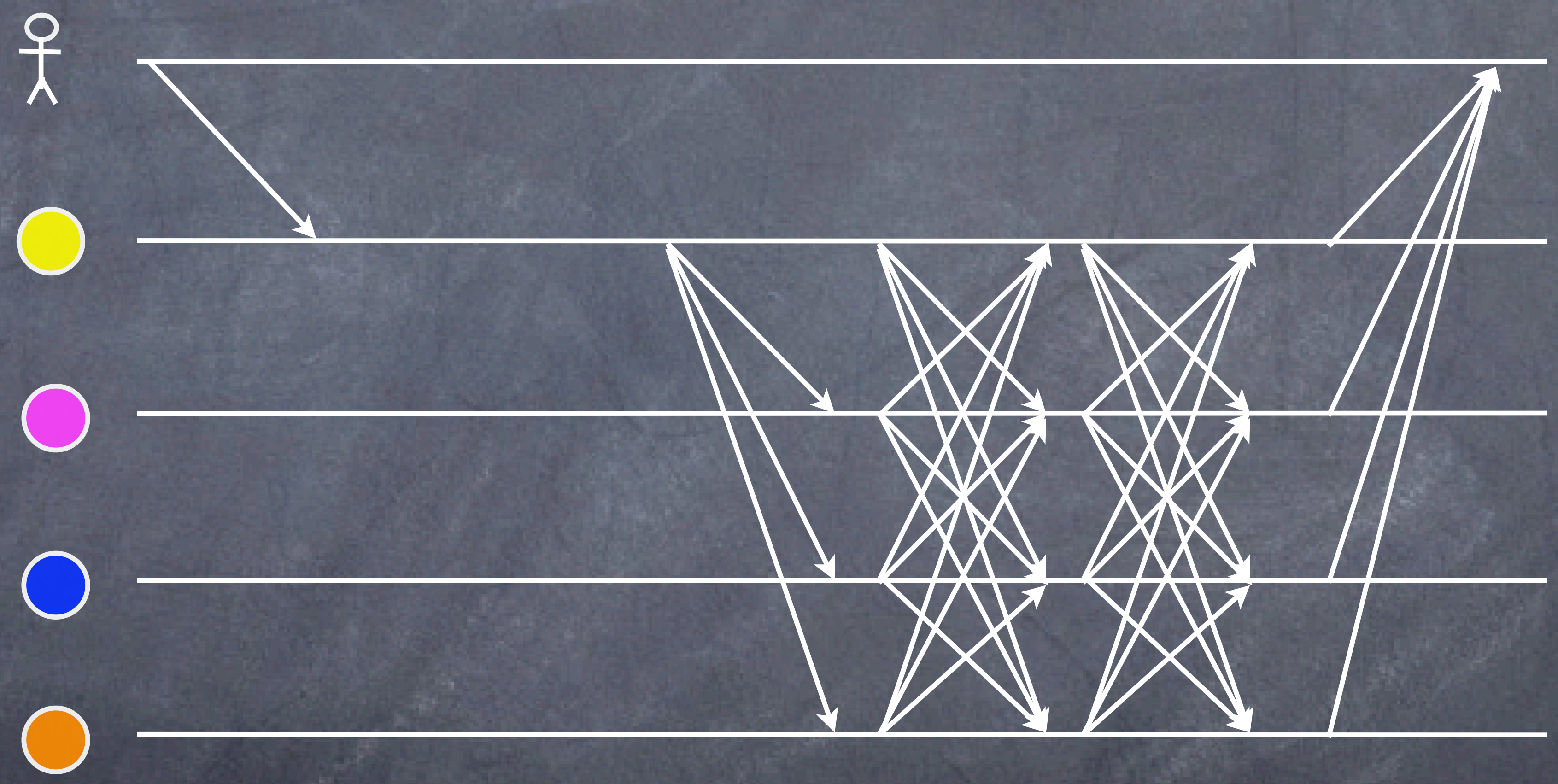
Big MAC Attack



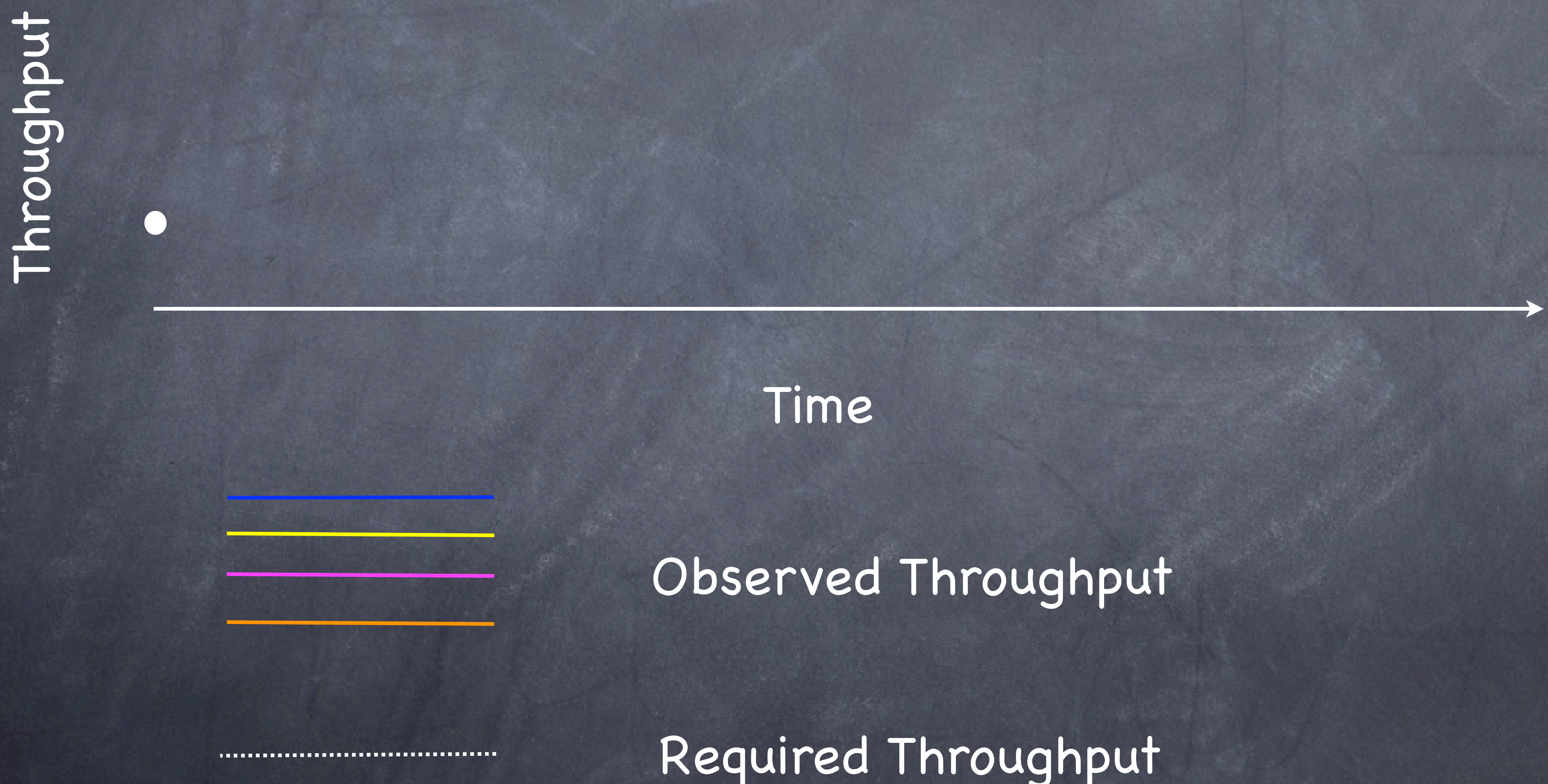
Slow Primary

Slow Primary

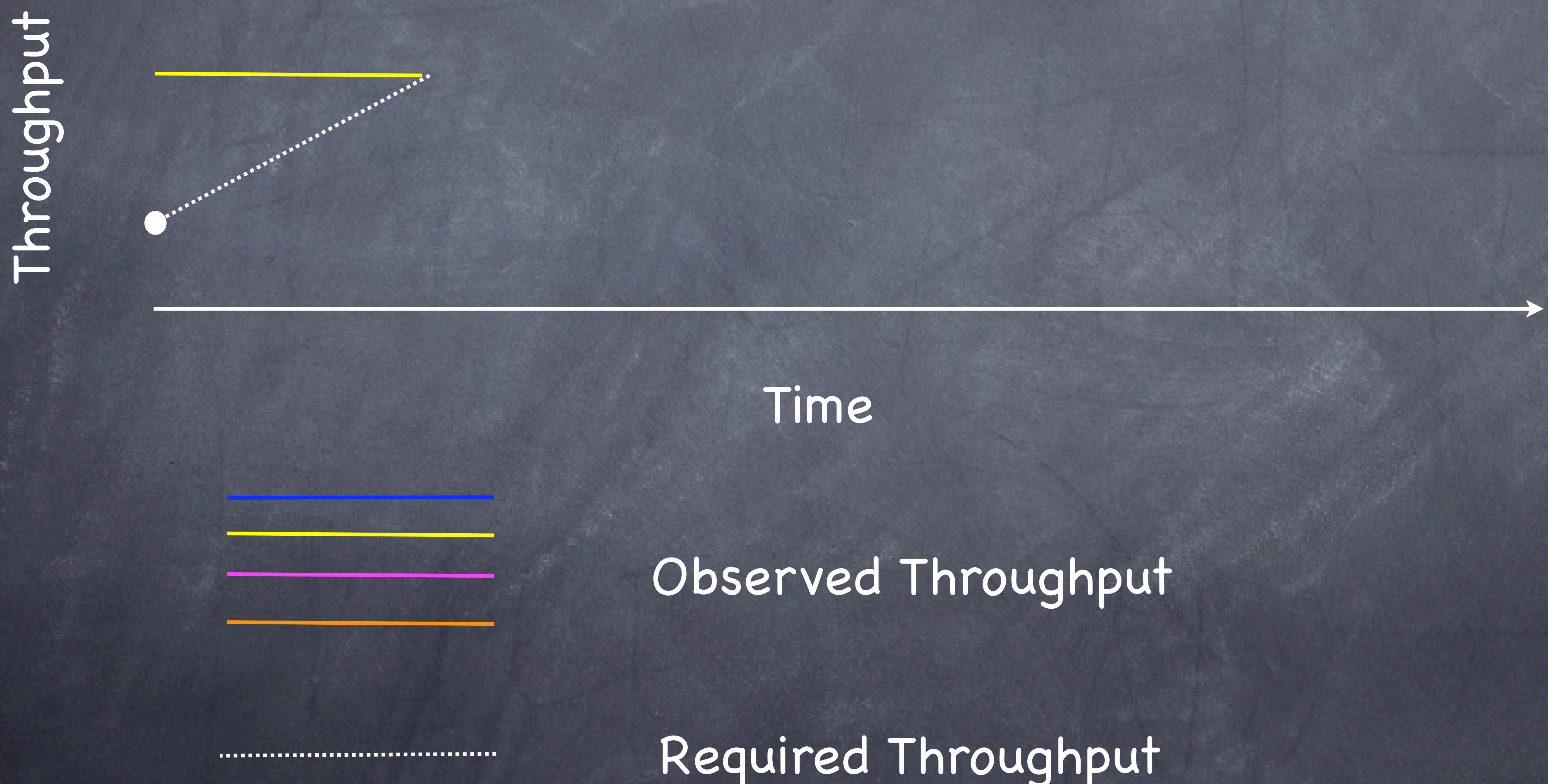
Slow Primary



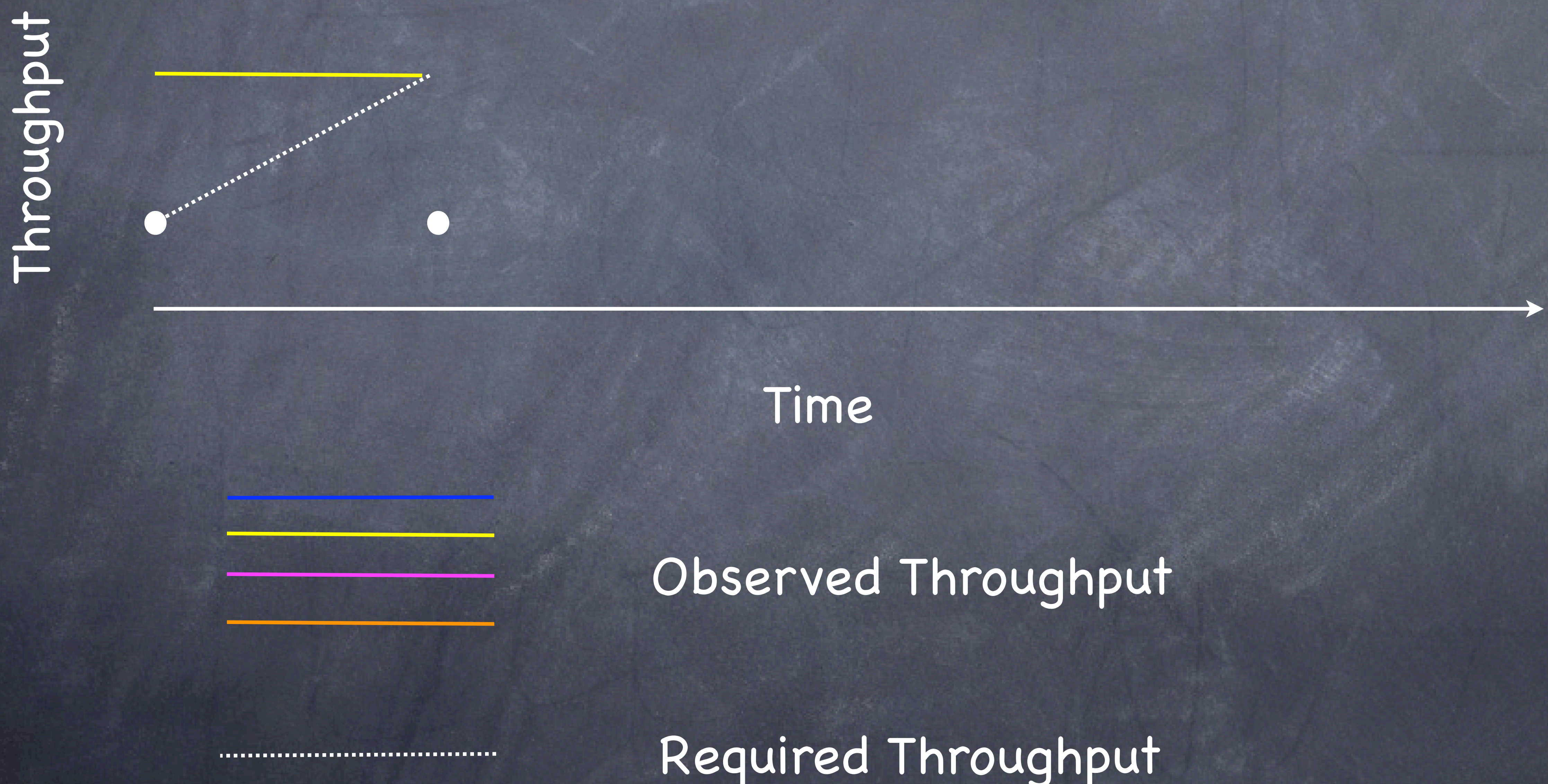
Adaptive View Changes



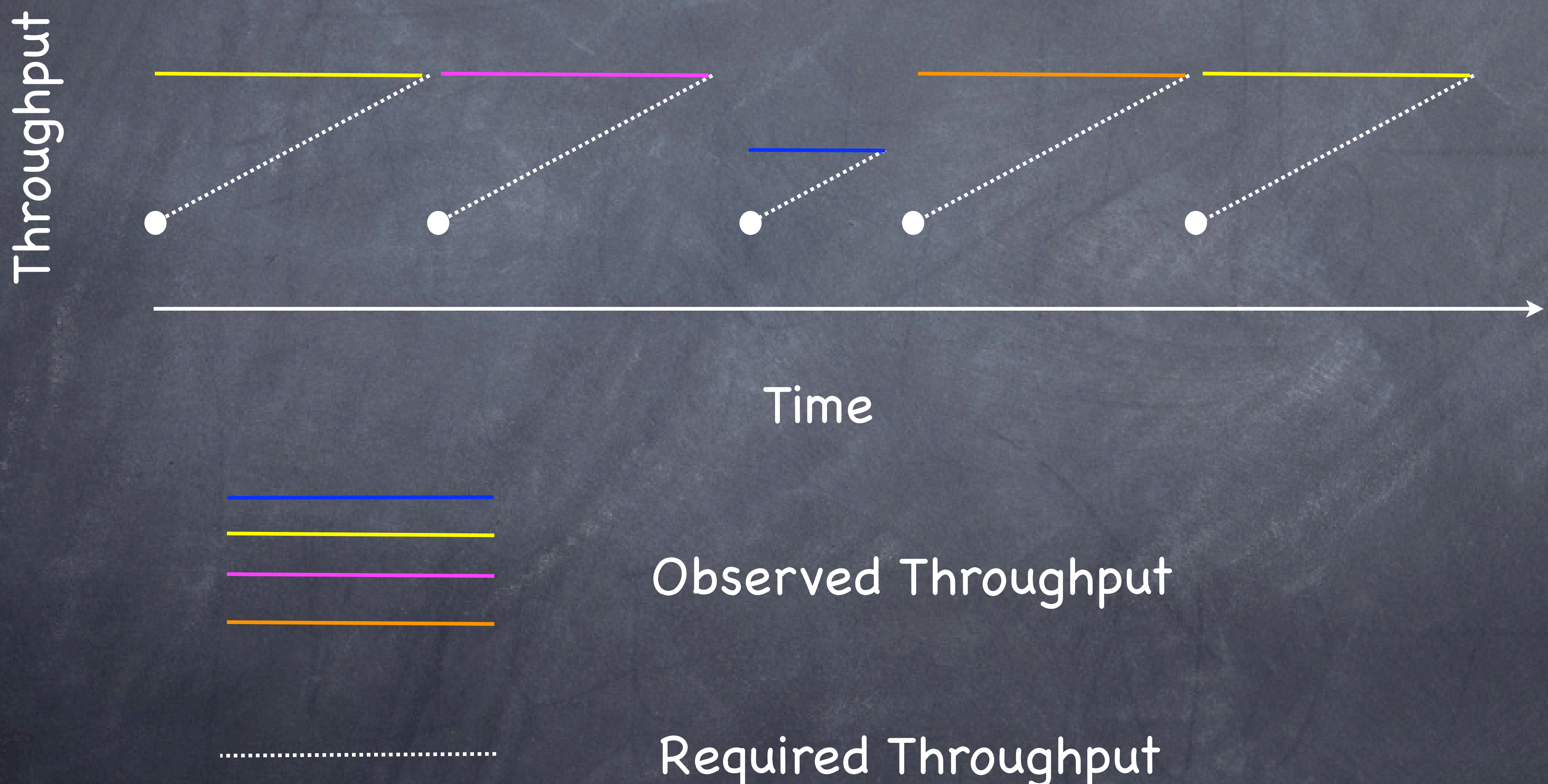
Adaptive View Changes



Adaptive View Changes



Adaptive View Changes



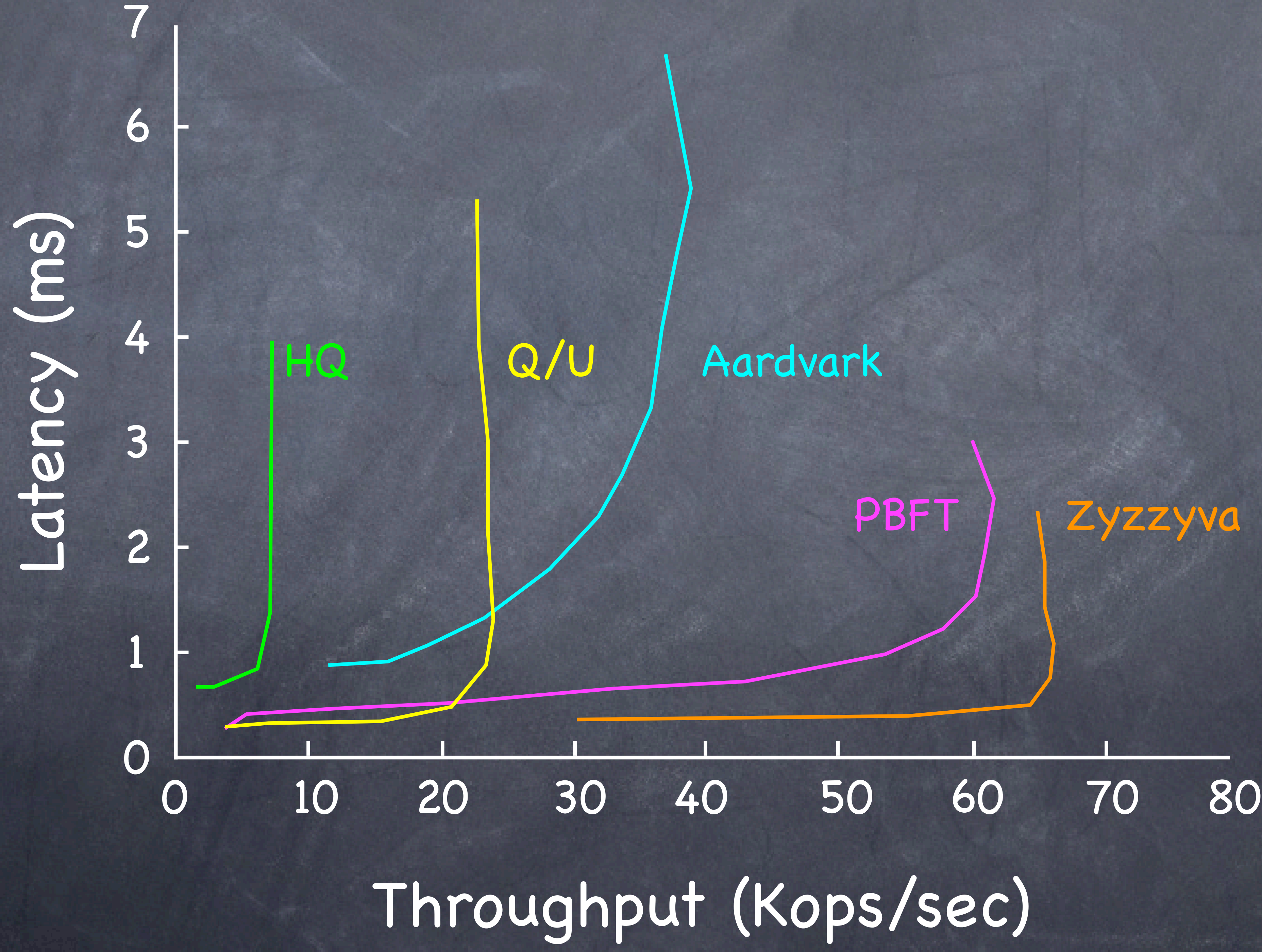
Implementation details

- Sign client requests
- Adaptive view change
- Separate network channels
- Fair scheduling
  - clients -v- replicas
  - replicas -v- replicas
- Exploit multicore architectures

Outline

- Robust BFT: The case for a new goal
- Aardvark: Designing for RBFT
- Evaluation: RBFT can work

Throughput -v- Latency



Aardvark, Incrementally

	MAC Client Request	Sign Client Request	Adaptive View Change
PBFT	62k	30k	-
Aardvark	58k	39k	39k

Performance with failures

- Byzantine failures are arbitrary
- Good faith effort

Big MAC Attack

	Peak	Faulty Client
PBFT	62k	0
Q/U	24k	0
HQ	7.6k	-
Zyzzyva	65k	0
Aardvark	39k	39k

Slow Primary

	Peak	1ms delay	10ms delay	100ms delay
PBFT	62k	5k	5k	1k
Zyzzyva	65k	28k	5k	crash
Aardvark	39k	38k	37k	38k

Summary

- RBFT: a new goal for BFT systems
- Aardvark: rejecting conventional wisdom
- Evaluation: it works!