Distributed Algorithms - Paper presentation

18/12/2018

ByzCast: Byzantine Fault-Tolerant Atomic Multicast

Alessia Ruggeri & Francesco Saverio Zuppichini

1.
What questions does the paper answer?

1. What questions does the paper answer?

The paper present **ByzCast**, the first Byzantine Fault-Tolerant atomic multicast.

1. What questions does the paper answer?

Improve existing protocol for atomic multicast by integrating it with existing BTF abstractions

Improve the ability to handle malicious behaviours

Improve the scalability of performances

2.

What are the main contributions of the paper?

2. What are the main contributions of the paper?

Present a partially genuine atomic multicast protocol that builds on multiple instances of atomic broadcast

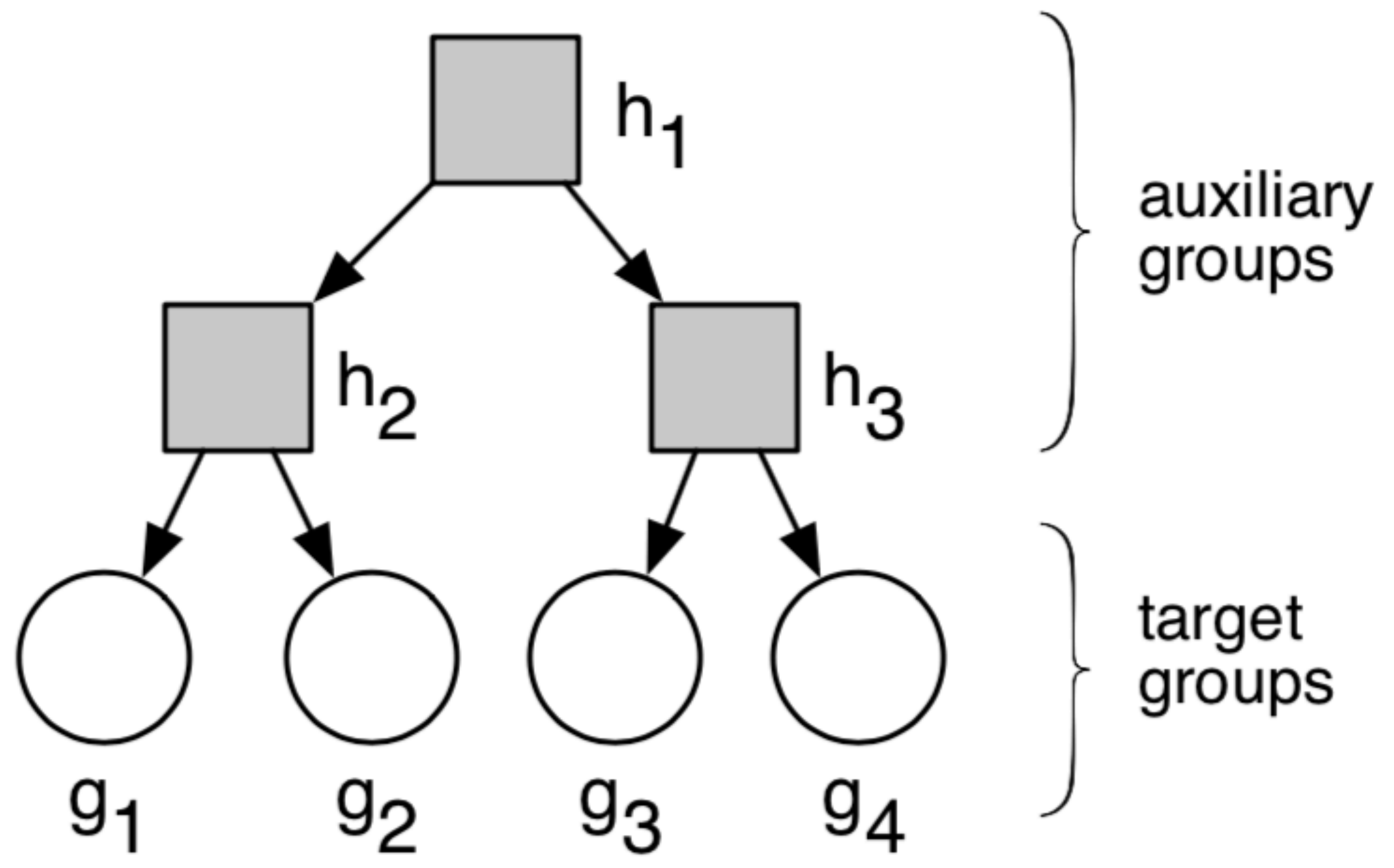
Define the problem of building an overlay tree as an optimisation problem

Describe a prototype of ByzCast

Provide an experimental evaluation of ByzCast

Partially genuine atomic multicast protocol

Protocol:



Target groups	$\Gamma = \{g_1, \ldots, g_m\}$
Auxiliary groups	$\Lambda = \{h_1, \ldots, h_n\}$

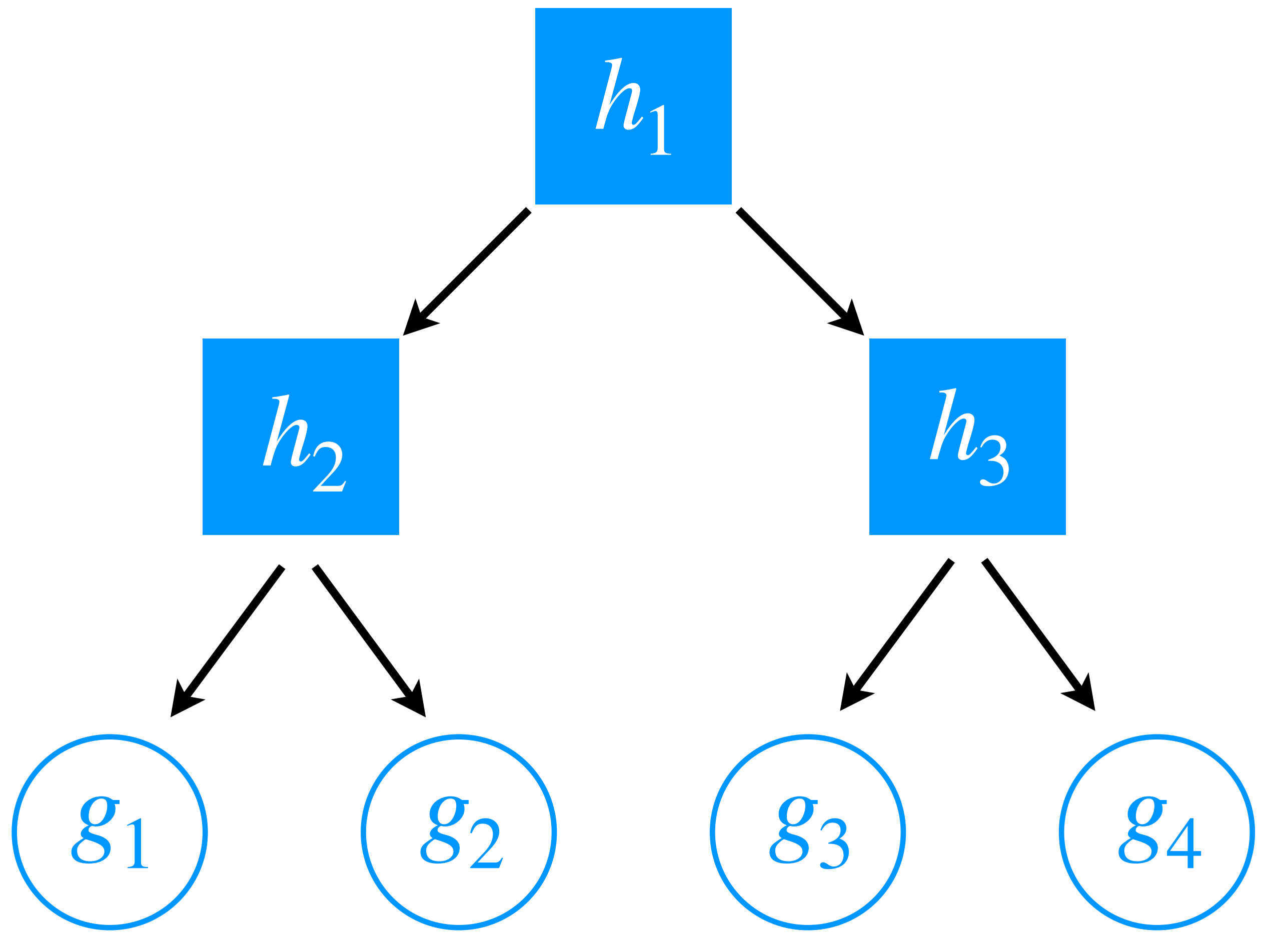
Each group *x* implements a **FIFO atomic broadcast**

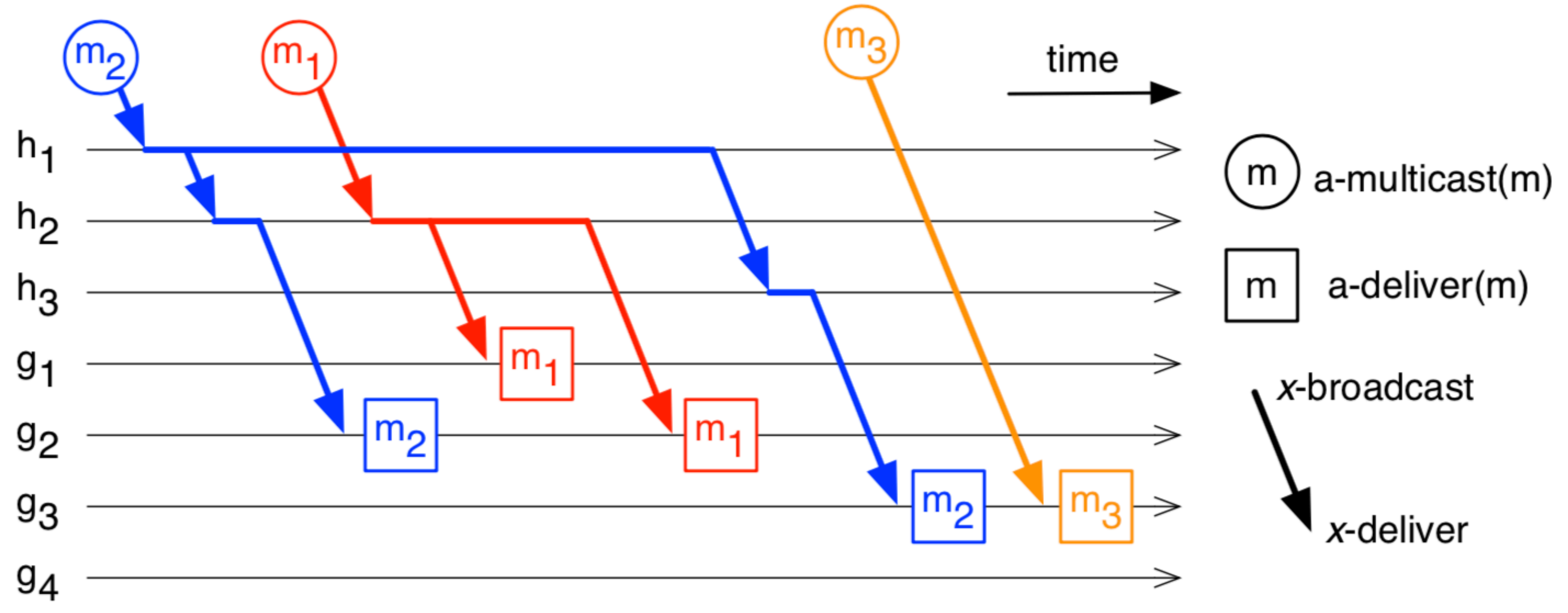
reach(x)	set of target groups that can be reached from *x* by walking down the tree
lca(m.dst)	lowest common ancestor

Partially genuine atomic multicast protocol

Alg. 1 ByzCast

```
1: Initialization
2:     $\mathcal{T}$ is an overlay tree with groups $\Gamma \cup \Lambda$
3:     $A\text{-}delivered \leftarrow \emptyset$
4: To a-multicast message $m$:
5:     $x_0 \leftarrow lca(m.dst)$          {lowest common ancestor of $m.dst$}
6:     $x_0$-broadcast$(m)$
7: Each server process $p$ in group $x_k$ executes as follows:
8:     when $x_k$-deliver$(m)$
9:         if $k = 0$ or $x_k$-delivered $m$ $(f+1)$ times then
10:            for each $x_{k+1} \in children(x_k)$ such that
                       $m.dst \cap reach(x_{k+1}) \neq \emptyset$ do
11:                $x_{k+1}$-broadcast$(m)$
12:            if $x_k \in m.dst$ and $m \notin A\text{-}delivered$ then
13:                a-deliver$(m)$
14:                $A\text{-}delivered \leftarrow A\text{-}delivered \cup \{m\}$
```

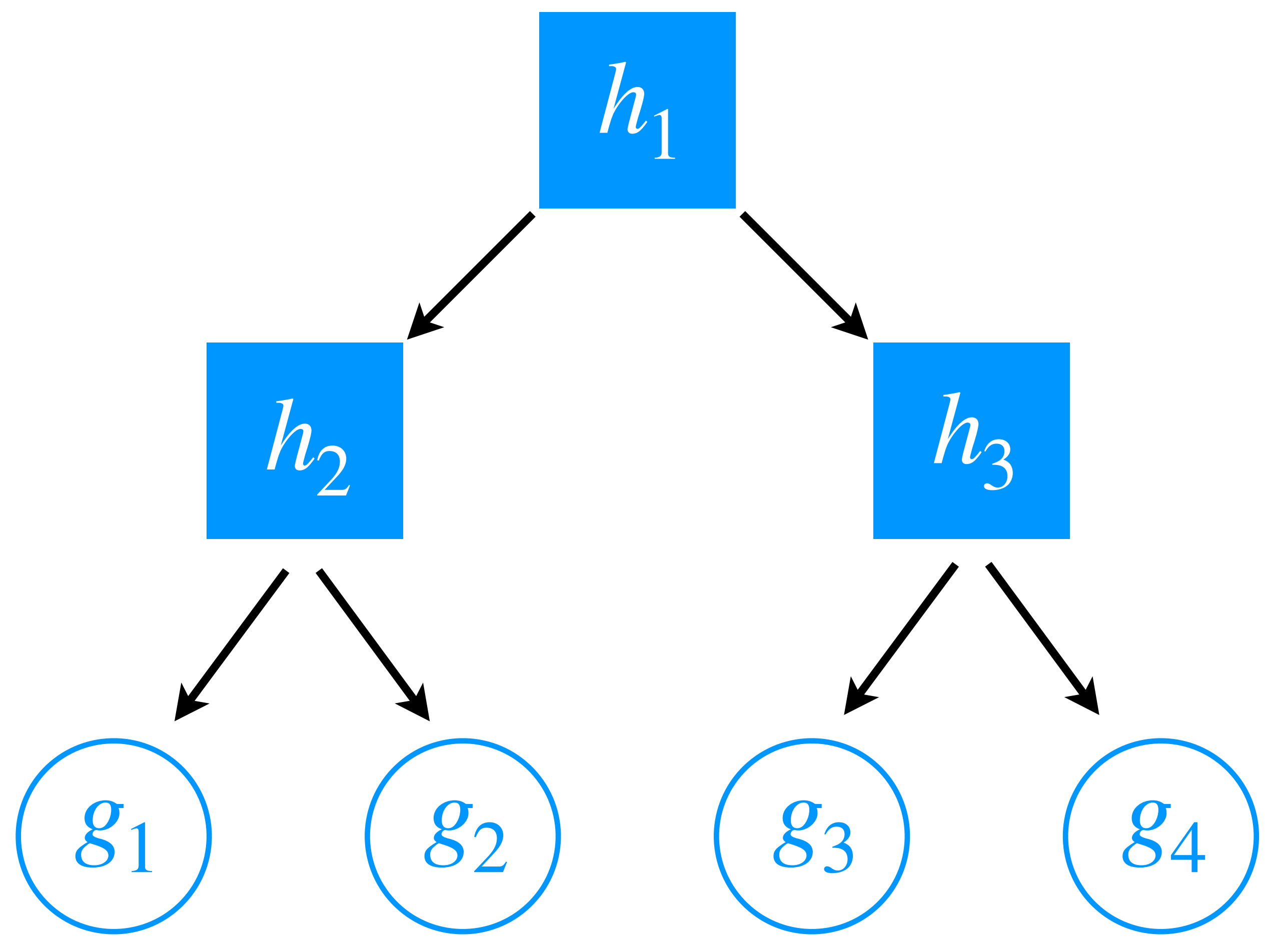


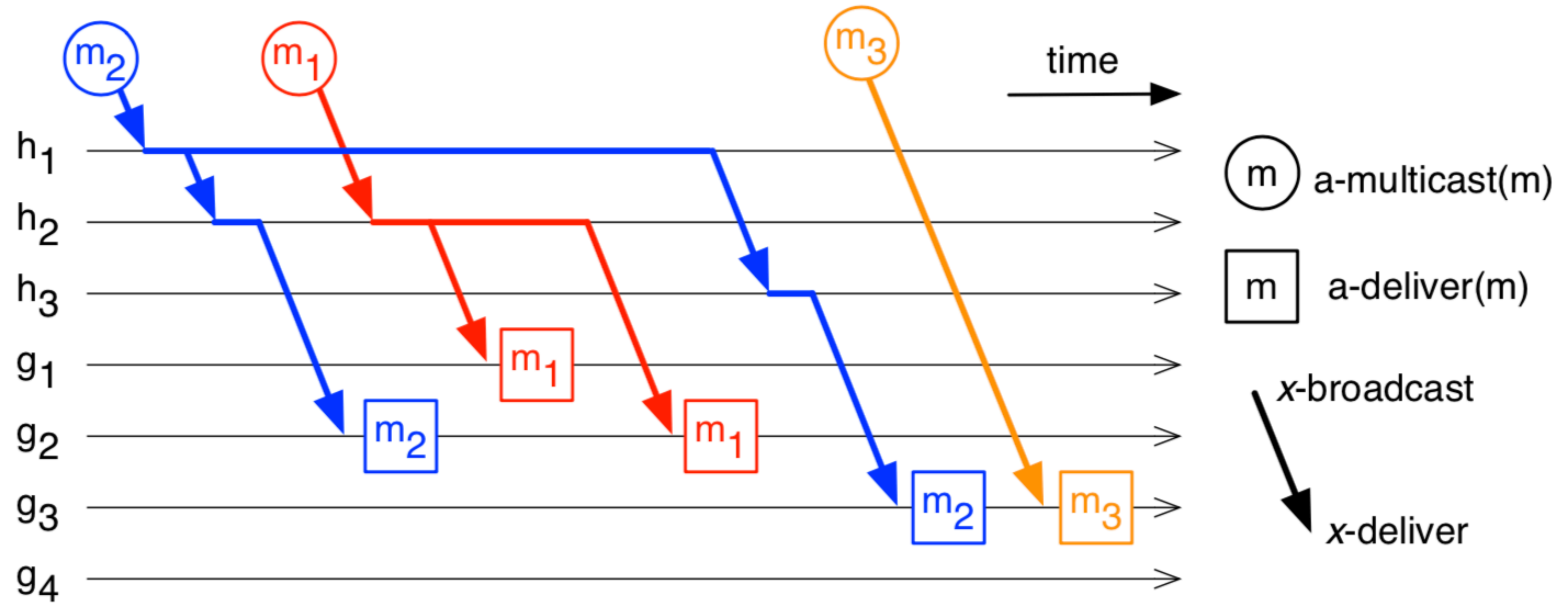


Partially genuine atomic multicast protocol

Alg. 1 ByzCast

```
1:  Initialization
2:    $\mathcal{T}$ is an overlay tree with groups $\Gamma \cup \Lambda$
3:    $A\text{-}delivered \leftarrow \emptyset$
4:  To a-multicast message $m$:
5:    $x_0 \leftarrow lca(m.dst)$            {lowest common ancestor of $m.dst$}
6:    $x_0$-broadcast$(m)$
7:  Each server process $p$ in group $x_k$ executes as follows:
8:    when $x_k$-deliver$(m)$
9:      if $k = 0$ or $x_k$-delivered $m$ $(f+1)$ times then
10:       for each $x_{k+1} \in children(x_k)$ such that
                $m.dst \cap reach(x_{k+1}) \neq \emptyset$ do
11:         $x_{k+1}$-broadcast$(m)$
12:       if $x_k \in m.dst$ and $m \notin A\text{-}delivered$ then
13:         a-deliver$(m)$
14:         $A\text{-}delivered \leftarrow A\text{-}delivered \cup \{m\}$
```





Partially genuine atomic multicast protocol

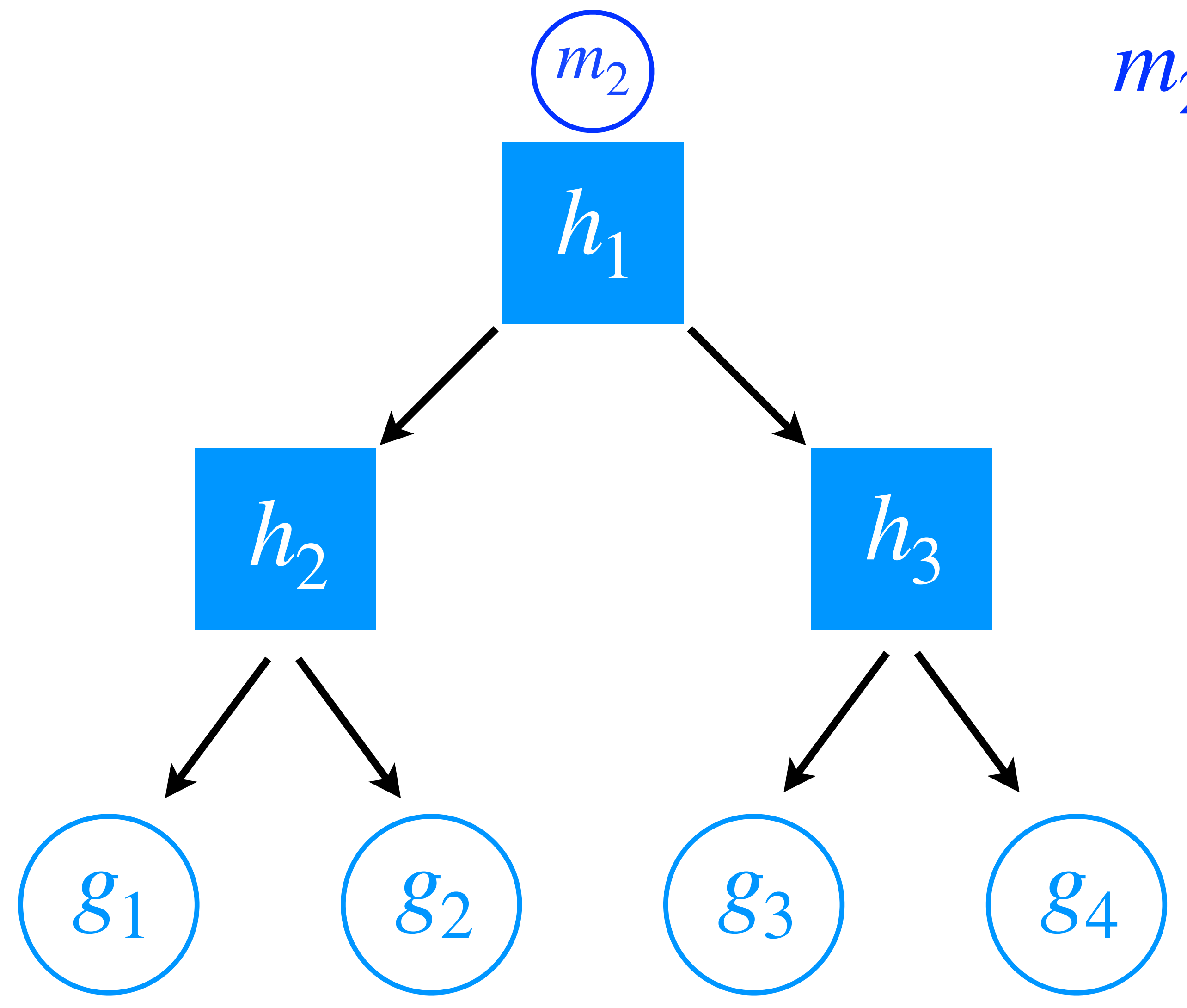
$m_2 . dst = \{g_2, g_3\}$

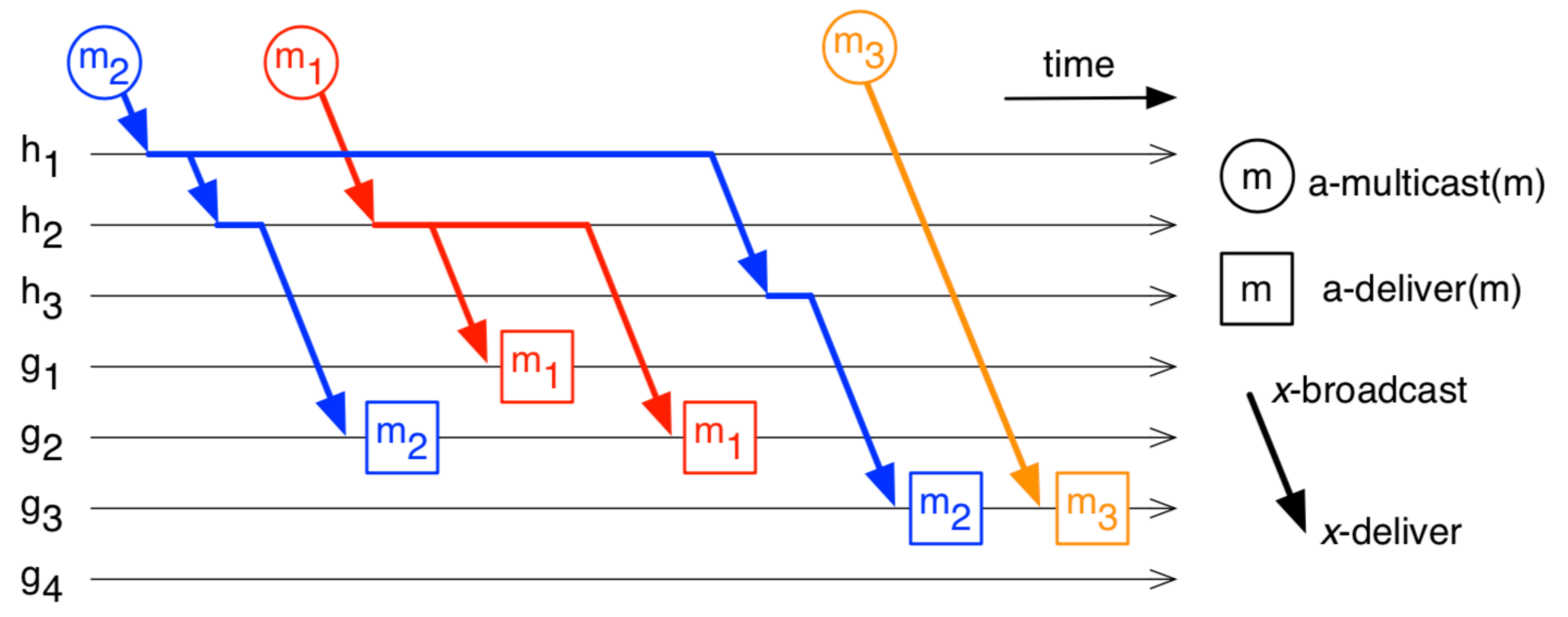
Alg. 1 ByzCast

```
1:  Initialization
2:     $\mathcal{T}$ is an overlay tree with groups $\Gamma \cup \Lambda$
3:     $A\text{-}delivered \leftarrow \emptyset$

4:  To a-multicast message $m$:
5:     $x_0 \leftarrow lca(m.dst)$          {lowest common ancestor of $m.dst$}
6:     $x_0$-broadcast$(m)$

7:  Each server process $p$ in group $x_k$ executes as follows:
8:     **when** $x_k$-deliver$(m)$
9:        **if** $k = 0$ **or** $x_k$-delivered $m$ $(f+1)$ times **then**
10:          **for each** $x_{k+1} \in children(x_k)$ such that
                          $m.dst \cap reach(x_{k+1}) \neq \emptyset$ **do**
11:             $x_{k+1}$-broadcast$(m)$
12:          **if** $x_k \in m.dst$ **and** $m \notin A\text{-}delivered$ **then**
13:             a-deliver$(m)$
14:             $A\text{-}delivered \leftarrow A\text{-}delivered \cup \{m\}$
```



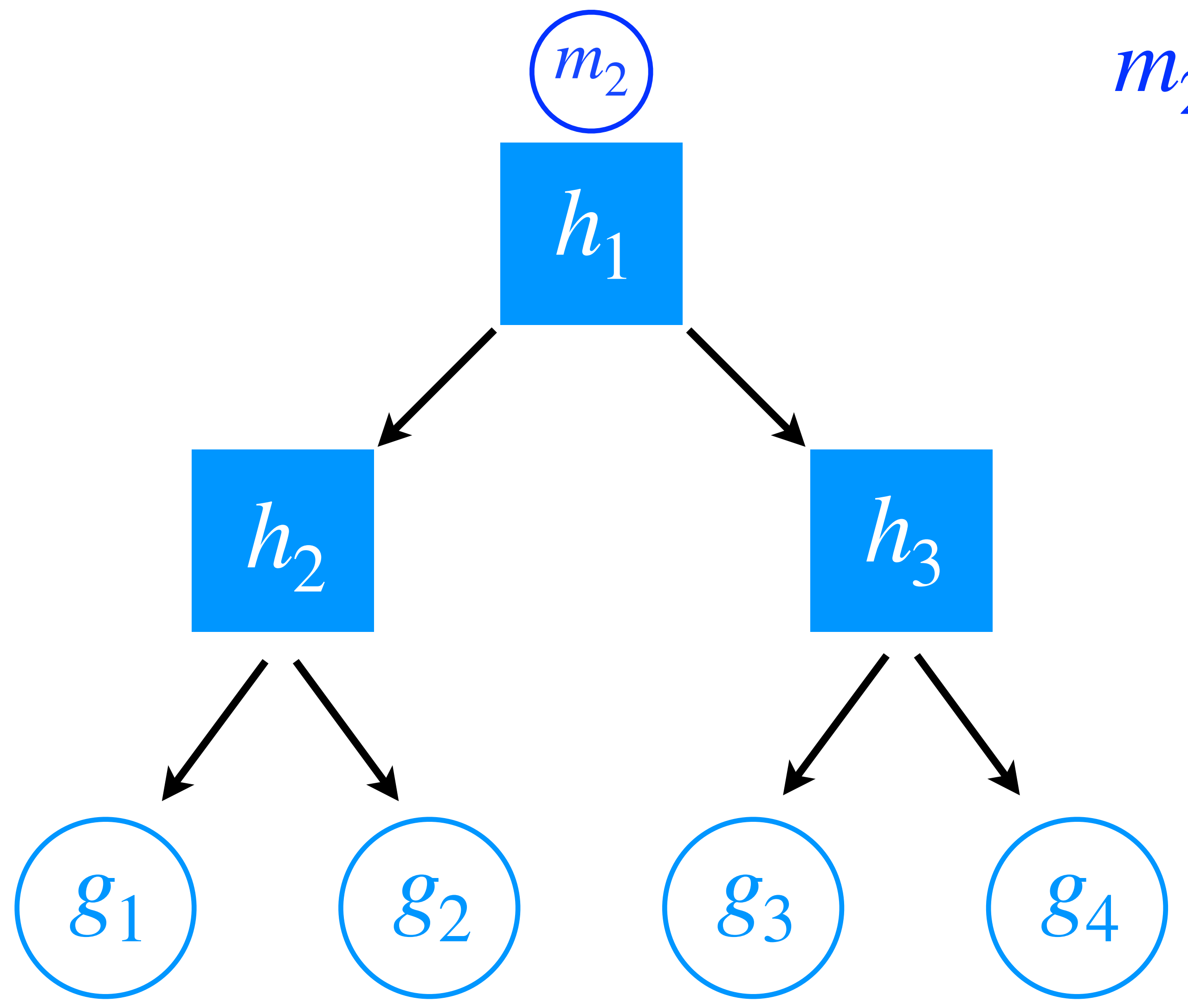


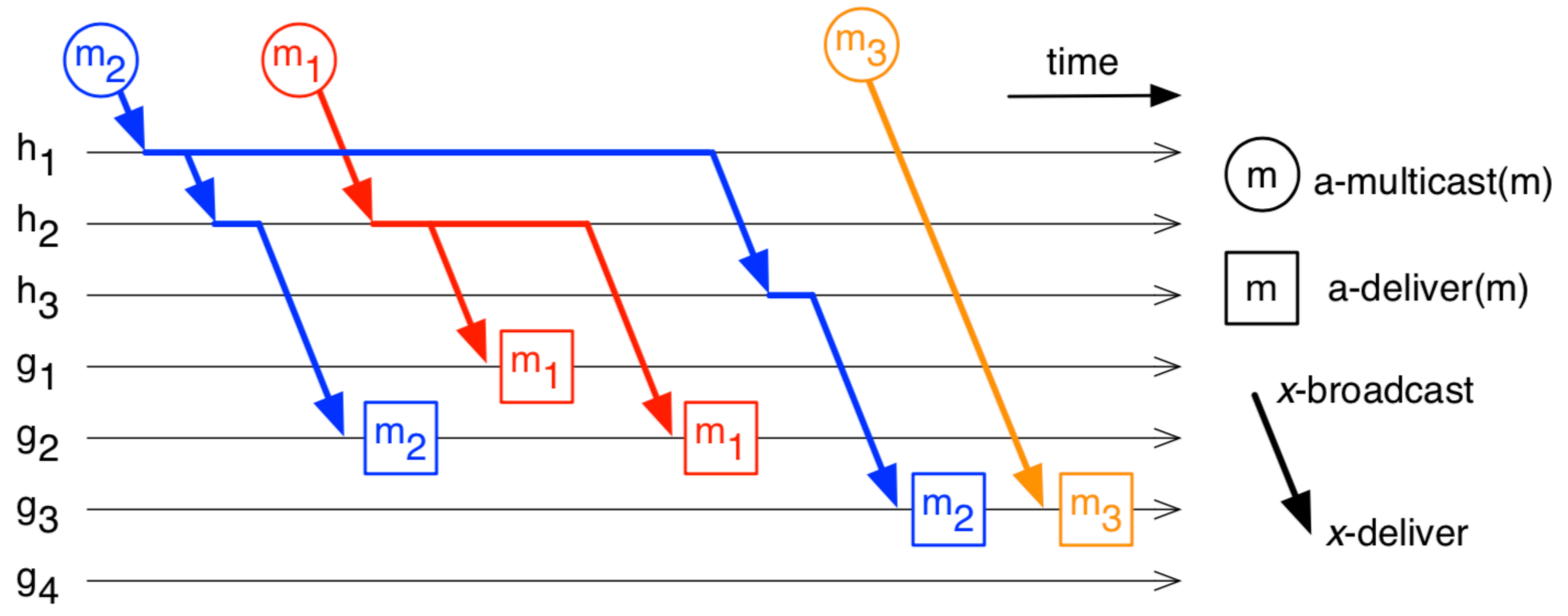
Partially genuine atomic multicast protocol

Alg. 1 ByzCast

```
1:  Initialization
2:     $\mathcal{T}$ is an overlay tree with groups $\Gamma \cup \Lambda$
3:     $A\text{-}delivered \leftarrow \emptyset$
4:  To a-multicast message $m$:
5:     $x_0 \leftarrow lca(m.dst)$        {lowest common ancestor of $m.dst$}
6:     $x_0$-broadcast$(m)$
7:  Each server process $p$ in group $x_k$ executes as follows:
8:     when $x_k$-deliver$(m)$
9:        if $k = 0$ or $x_k$-delivered $m$ $(f+1)$ times then
10:          for each $x_{k+1} \in children(x_k)$ such that
                      $m.dst \cap reach(x_{k+1}) \neq \emptyset$ do
11:             $x_{k+1}$-broadcast$(m)$
12:          if $x_k \in m.dst$ and $m \notin A\text{-}delivered$ then
13:             a-deliver$(m)$
14:             $A\text{-}delivered \leftarrow A\text{-}delivered \cup \{m\}$
```

$m_2 . dst = \{g_2, g_3\}$

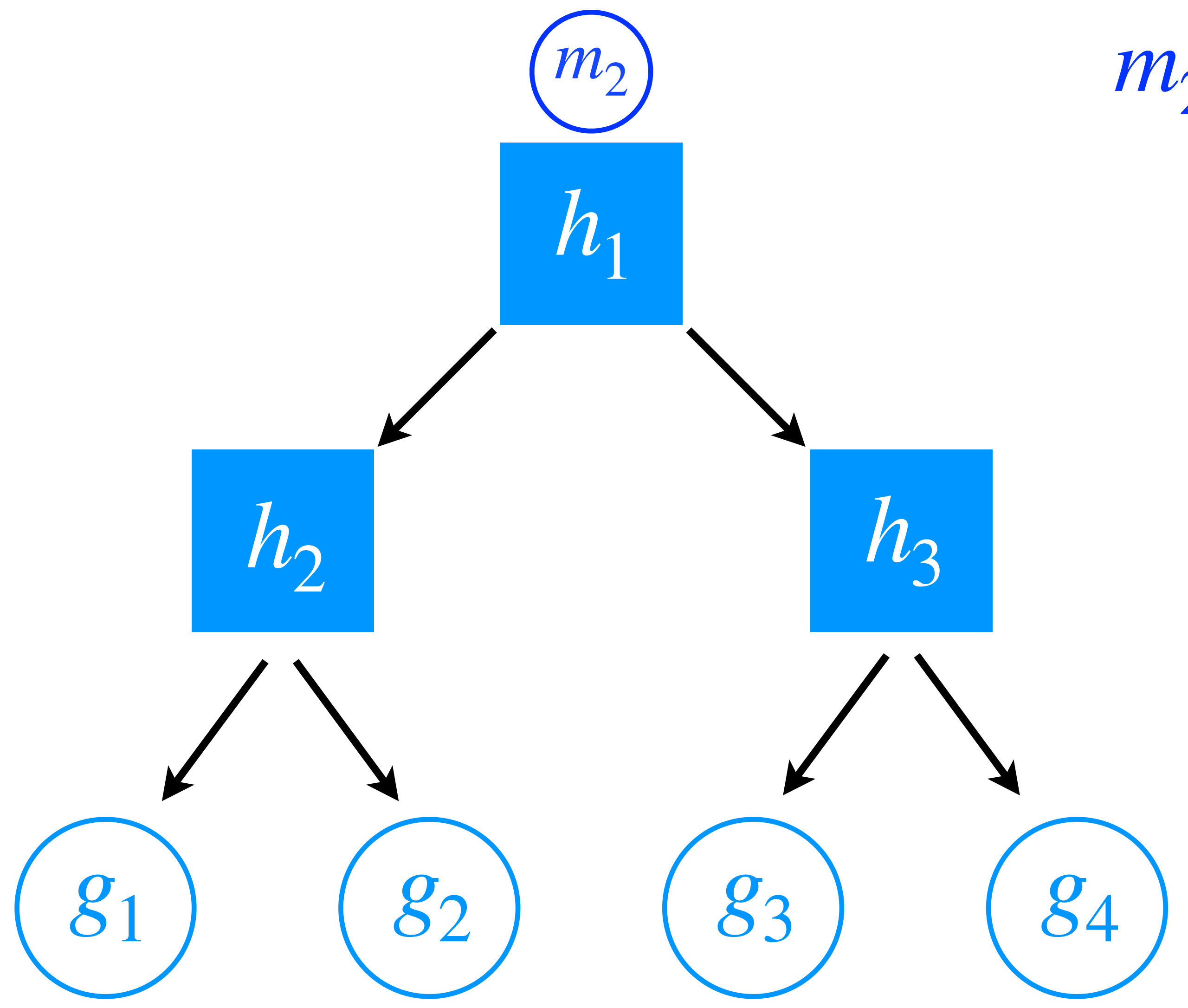




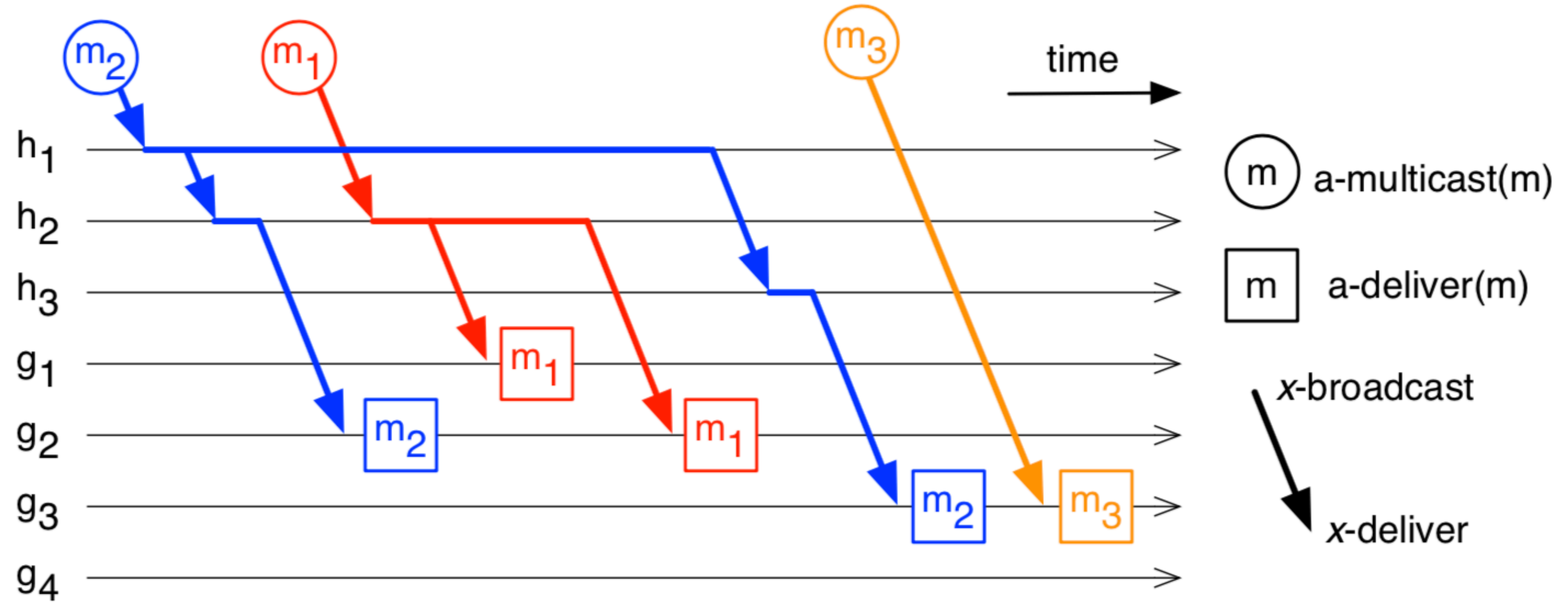
Partially genuine atomic multicast protocol

Alg. 1 ByzCast

```
1:  Initialization
2:      $\mathcal{T}$ is an overlay tree with groups $\Gamma \cup \Lambda$
3:      $A\text{-}delivered \leftarrow \emptyset$
4:  To a-multicast message $m$:
5:      $x_0 \leftarrow lca(m.dst)$          {lowest common ancestor of $m.dst$}
6:      $x_0$-broadcast($m$)
7:  Each server process $p$ in group $x_k$ executes as follows:
8:      when $x_k$-deliver($m$)
9:          if $k = 0$ or $x_k$-delivered $m$ $(f+1)$ times then
10:             for each $x_{k+1} \in children(x_k)$ such that
                        $m.dst \cap reach(x_{k+1}) \neq \emptyset$ do
11:                 $x_{k+1}$-broadcast($m$)
12:             if $x_k \in m.dst$ and $m \notin A\text{-}delivered$ then
13:                 a-deliver($m$)
14:                 $A\text{-}delivered \leftarrow A\text{-}delivered \cup \{m\}$
```



$m_2 . dst = \{g_2, g_3\}$

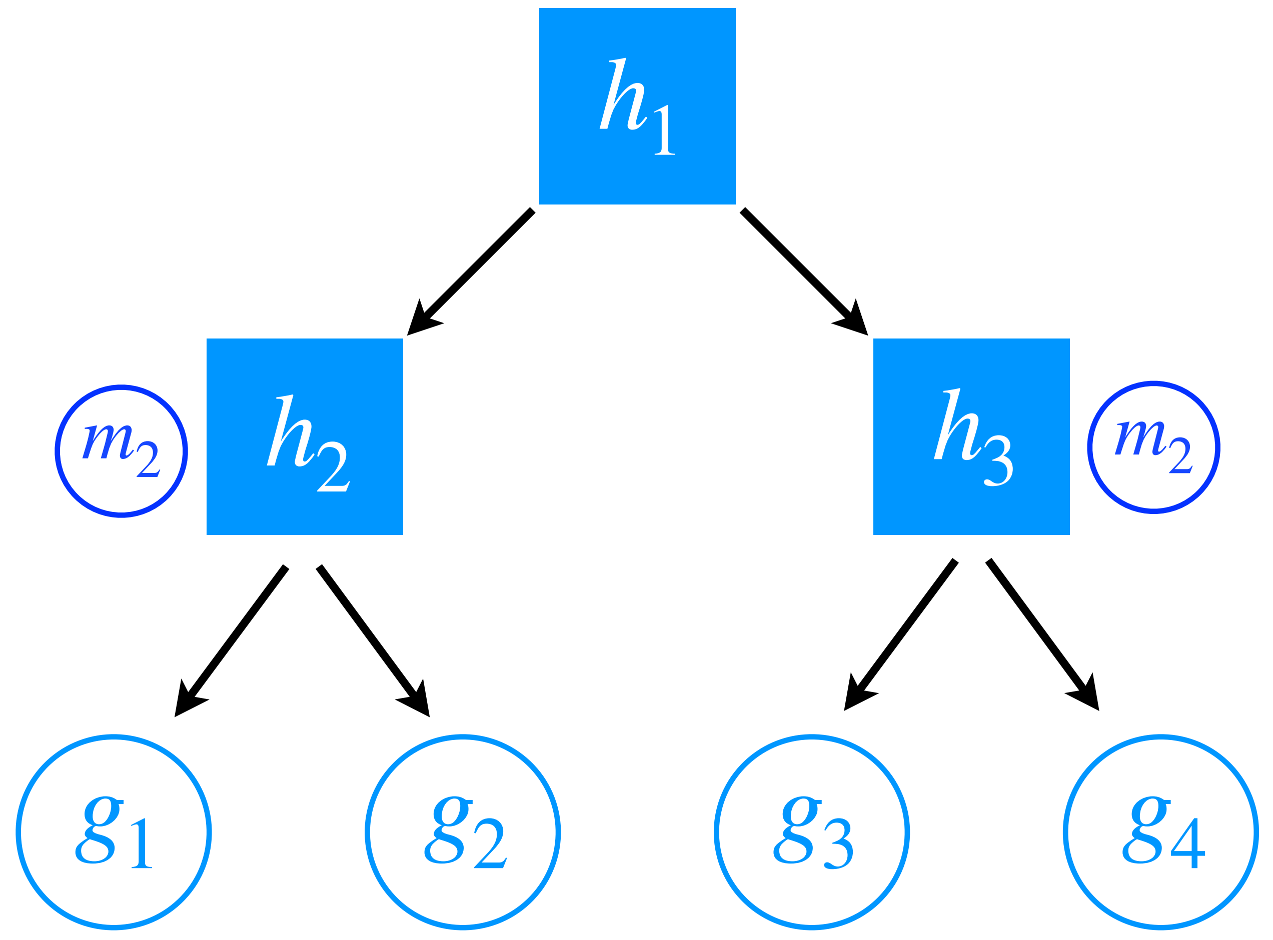


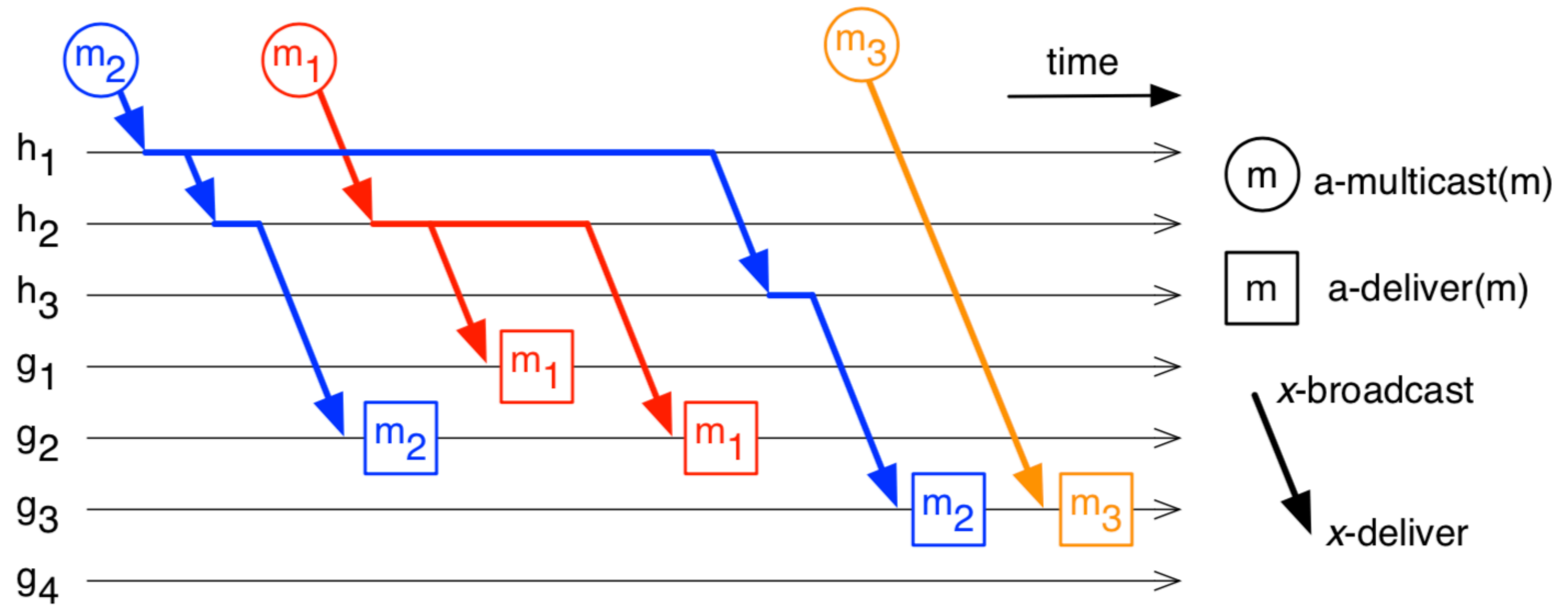
Partially genuine atomic multicast protocol

$m_2\,.\,dst = \{g_2, g_3\}$

Alg. 1 ByzCast

```
1:  Initialization
2:     $\mathcal{T}$ is an overlay tree with groups $\Gamma \cup \Lambda$
3:     $A\text{-}delivered \leftarrow \emptyset$
4:  To a-multicast message $m$:
5:     $x_0 \leftarrow lca(m.dst)$          {lowest common ancestor of $m.dst$}
6:     $x_0$-broadcast$(m)$
7:  Each server process $p$ in group $x_k$ executes as follows:
8:     when $x_k$-deliver$(m)$
9:        if $k = 0$ or $x_k$-delivered $m$ $(f+1)$ times then
10:          for each $x_{k+1} \in children(x_k)$ such that
                         $m.dst \cap reach(x_{k+1}) \neq \emptyset$ do
11:             $x_{k+1}$-broadcast$(m)$
12:          if $x_k \in m.dst$ and $m \notin A\text{-}delivered$ then
13:             a-deliver$(m)$
14:             $A\text{-}delivered \leftarrow A\text{-}delivered \cup \{m\}$
```





Partially genuine atomic multicast protocol

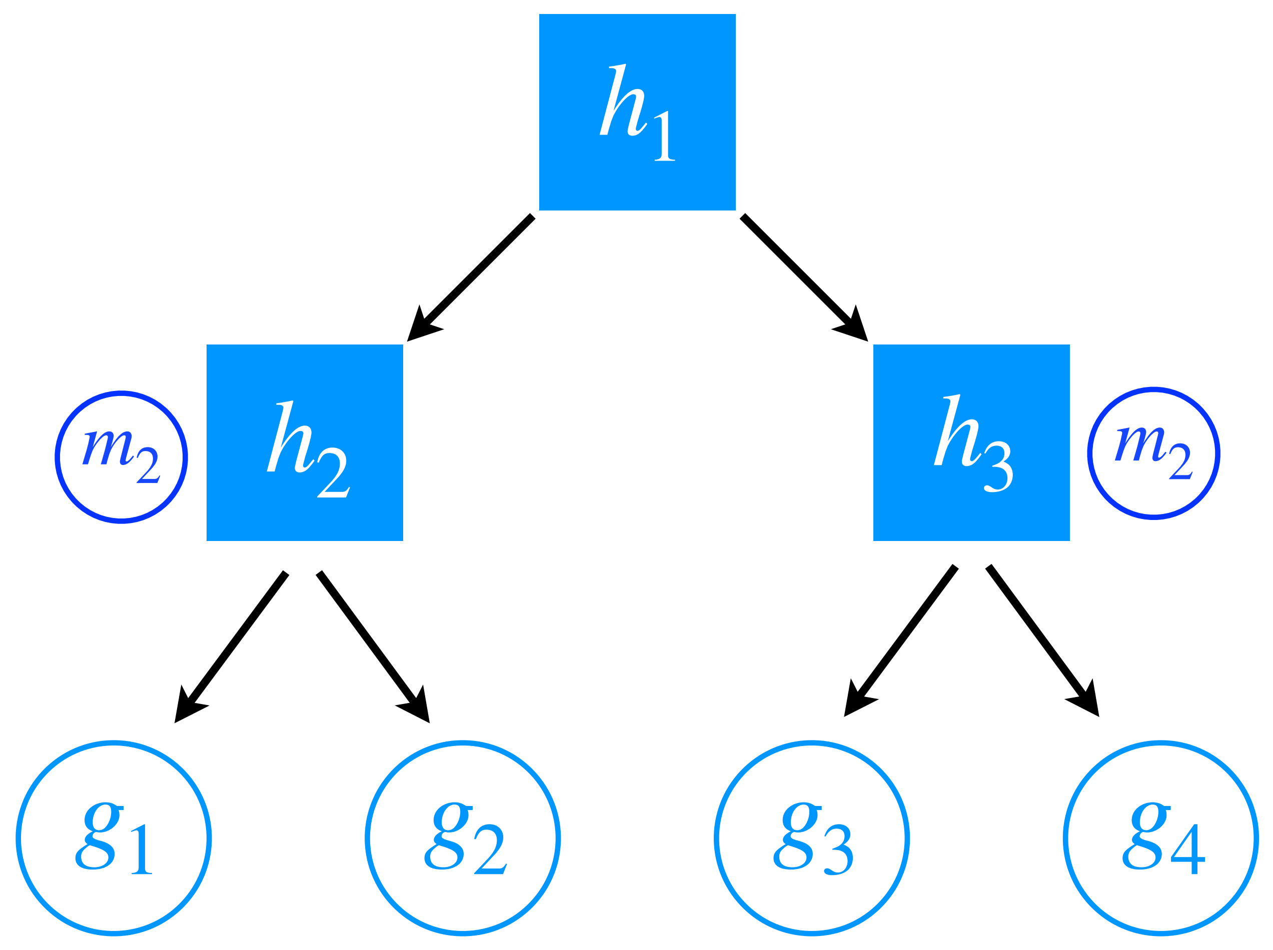
$m_2 . dst = \{g_2, g_3\}$

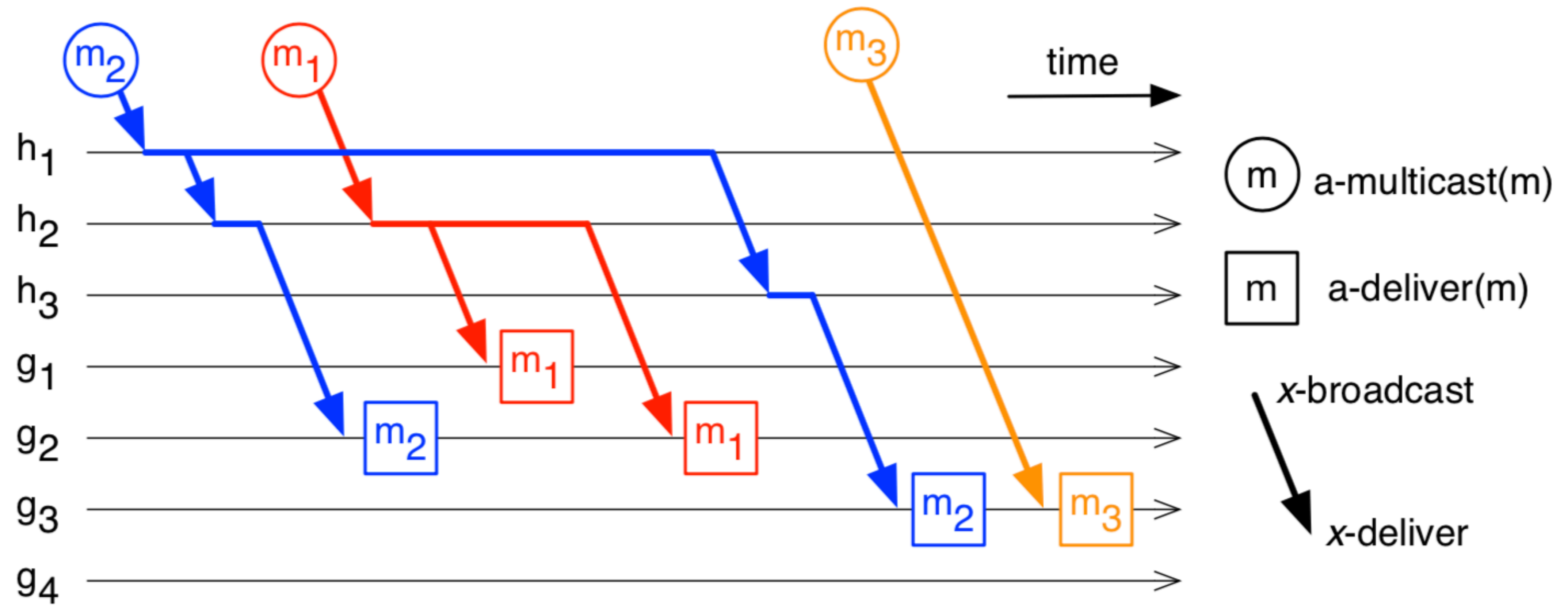
Alg. 1 ByzCast

```
1:  Initialization
2:     $\mathcal{T}$ is an overlay tree with groups $\Gamma \cup \Lambda$
3:     $A\text{-}delivered \leftarrow \emptyset$

4:  To a-multicast message $m$:
5:     $x_0 \leftarrow lca(m.dst)$          {lowest common ancestor of $m.dst$}
6:     $x_0$-broadcast$(m)$

7:  Each server process $p$ in group $x_k$ executes as follows:
8:     when $x_k$-deliver$(m)$
9:        if $k = 0$ or $x_k$-delivered $m$ $(f+1)$ times then
10:          for each $x_{k+1} \in children(x_k)$ such that
                      $m.dst \cap reach(x_{k+1}) \neq \emptyset$ do
11:             $x_{k+1}$-broadcast$(m)$
12:          if $x_k \in m.dst$ and $m \notin A\text{-}delivered$ then
13:             a-deliver$(m)$
14:             $A\text{-}delivered \leftarrow A\text{-}delivered \cup \{m\}$
```





Partially genuine atomic multicast protocol

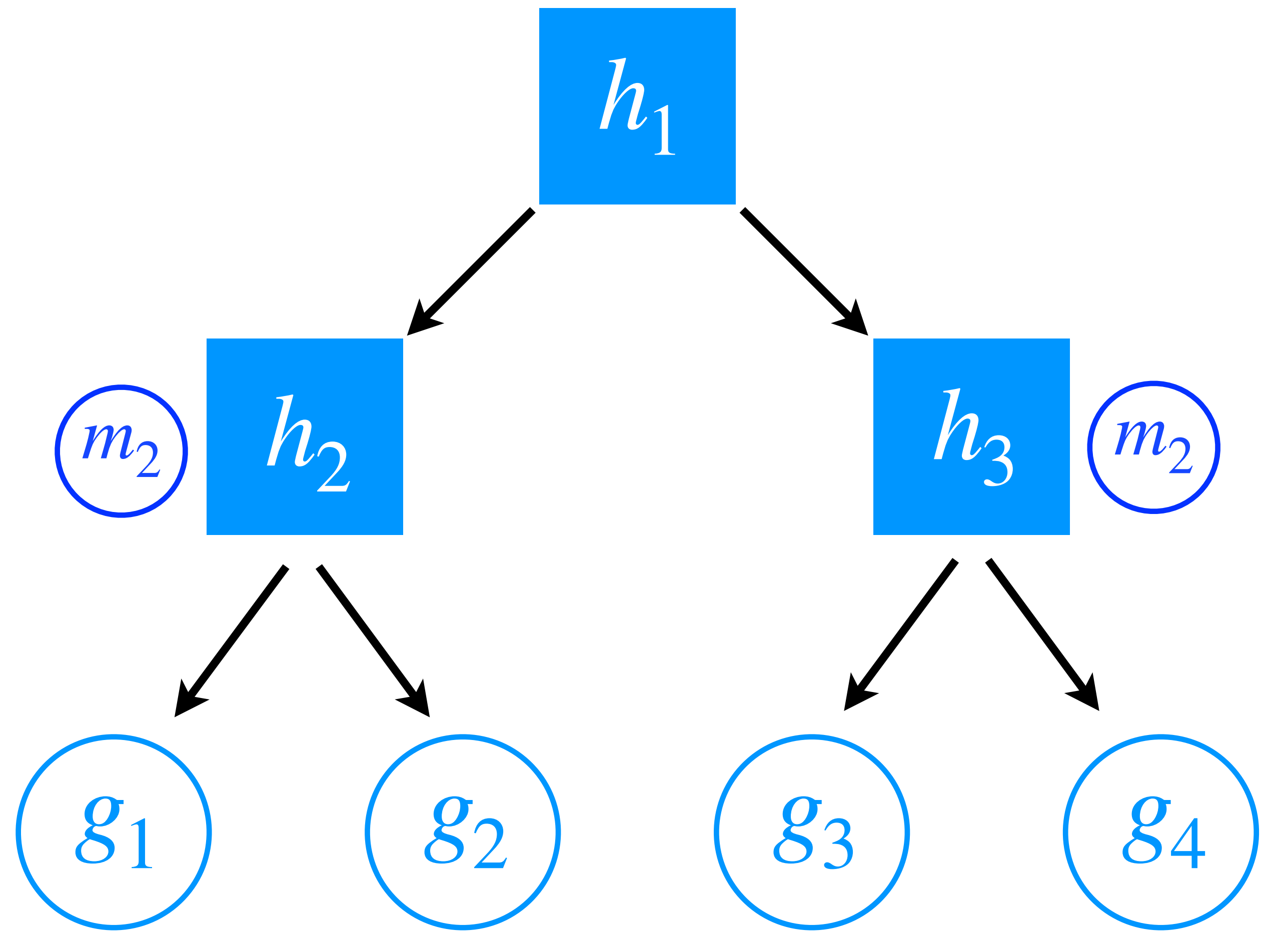
$m_2 . dst = \{g_2, g_3\}$

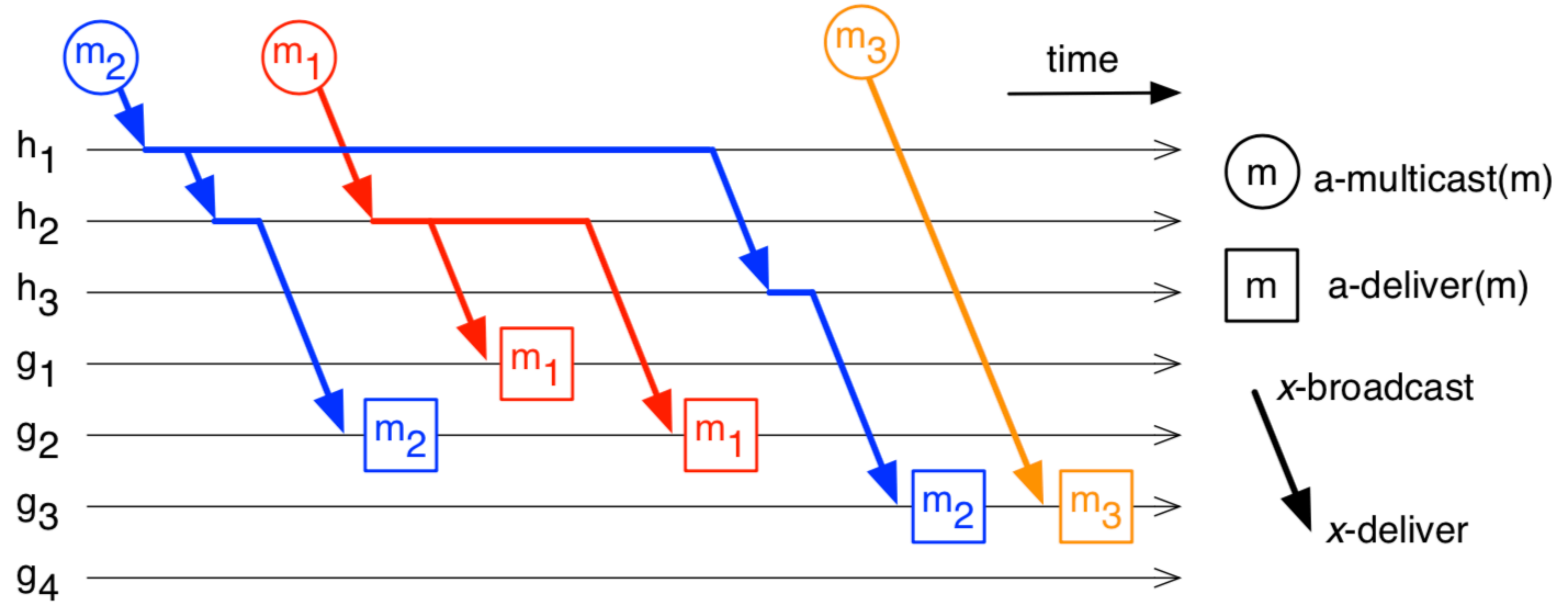
Alg. 1 ByzCast

```
1:  Initialization
2:      $\mathcal{T}$ is an overlay tree with groups $\Gamma \cup \Lambda$
3:      $A\text{-}delivered \leftarrow \emptyset$

4:  To a-multicast message $m$:
5:      $x_0 \leftarrow lca(m.dst)$            {lowest common ancestor of $m.dst$}
6:      $x_0$-broadcast$(m)$

7:  Each server process $p$ in group $x_k$ executes as follows:
8:      when $x_k$-deliver$(m)$
9:          if $k = 0$ or $x_k$-delivered $m$ $(f+1)$ times then
10:             for each $x_{k+1} \in children(x_k)$ such that
                        $m.dst \cap reach(x_{k+1}) \neq \emptyset$ do
11:                 $x_{k+1}$-broadcast$(m)$
12:             if $x_k \in m.dst$ and $m \notin A\text{-}delivered$ then
13:                 a-deliver$(m)$
14:                 $A\text{-}delivered \leftarrow A\text{-}delivered \cup \{m\}$
```



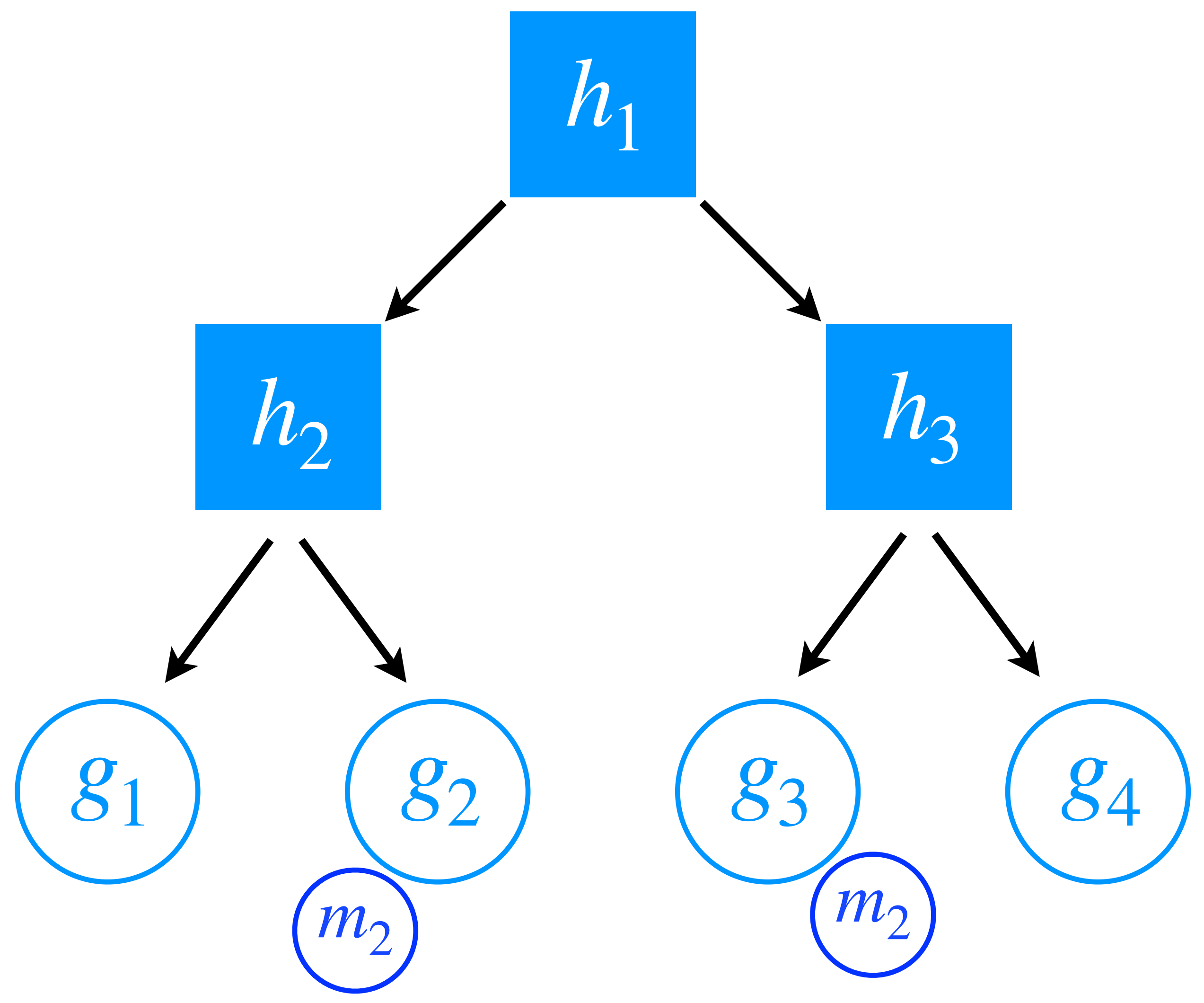


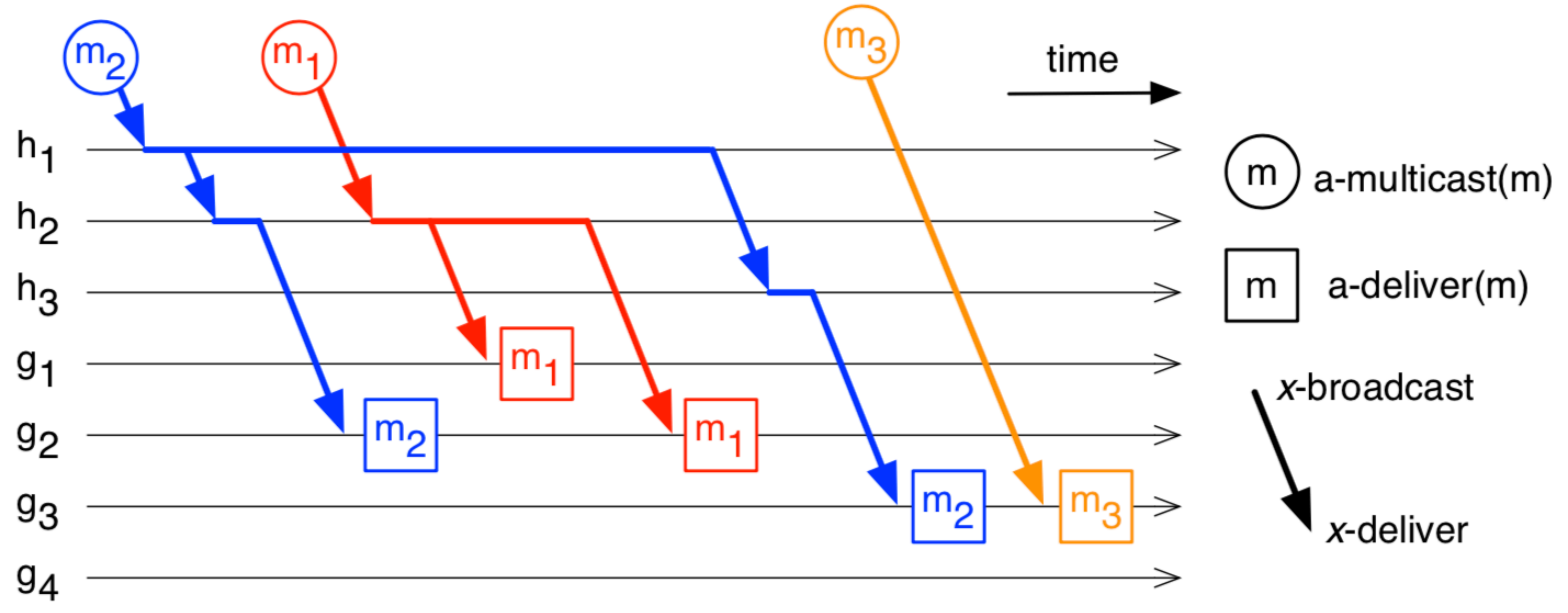
Partially genuine atomic multicast protocol

$m_2 . dst = \{g_2, g_3\}$

Alg. 1 ByzCast

```
1:  Initialization
2:     $\mathcal{T}$ is an overlay tree with groups $\Gamma \cup \Lambda$
3:     $A\text{-}delivered \leftarrow \emptyset$
4:  To a-multicast message $m$:
5:     $x_0 \leftarrow lca(m.dst)$          {lowest common ancestor of $m.dst$}
6:     $x_0$-broadcast($m$)
7:  Each server process $p$ in group $x_k$ executes as follows:
8:     when $x_k$-deliver($m$)
9:        if $k = 0$ or $x_k$-delivered $m$ $(f+1)$ times then
10:          for each $x_{k+1} \in children(x_k)$ such that
                   $m.dst \cap reach(x_{k+1}) \neq \emptyset$ do
11:             $x_{k+1}$-broadcast($m$)
12:          if $x_k \in m.dst$ and $m \notin A\text{-}delivered$ then
13:             a-deliver($m$)
14:             $A\text{-}delivered \leftarrow A\text{-}delivered \cup \{m\}$
```



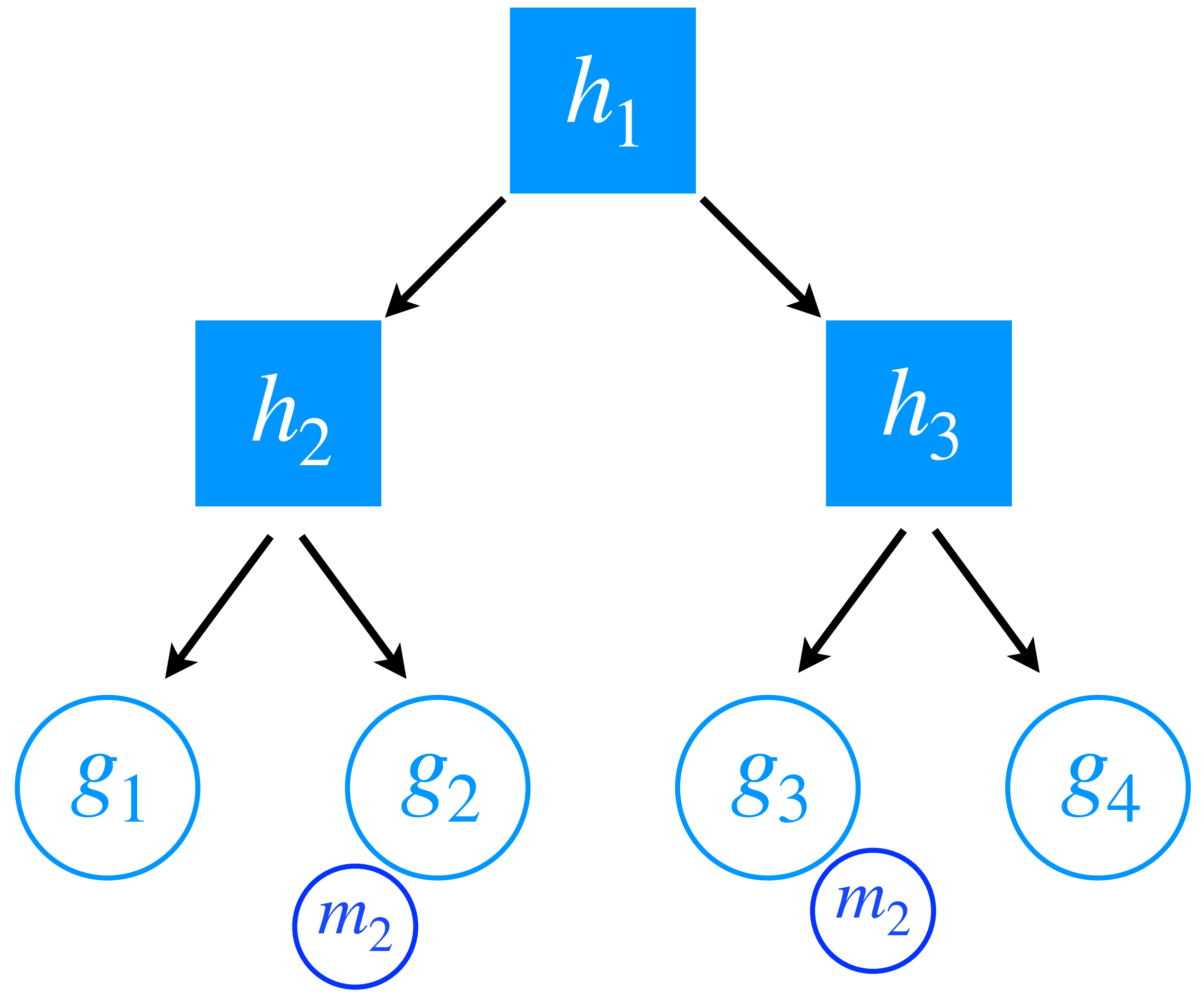


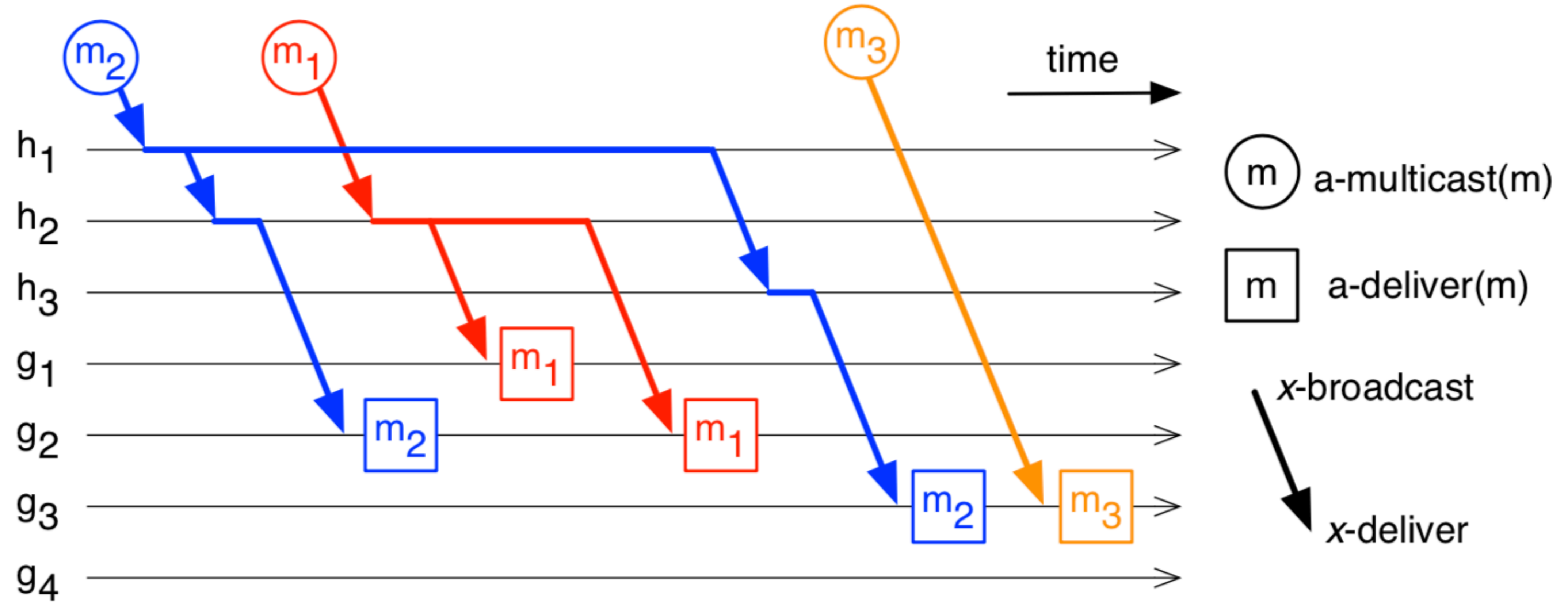
Partially genuine atomic multicast protocol

$m_2 . dst = \{g_2, g_3\}$

Alg. 1 ByzCast

```
1:  Initialization
2:      $\mathcal{T}$ is an overlay tree with groups $\Gamma \cup \Lambda$
3:      $A\text{-}delivered \leftarrow \emptyset$
4:  To a-multicast message $m$:
5:      $x_0 \leftarrow lca(m.dst)$          {lowest common ancestor of $m.dst$}
6:      $x_0$-broadcast$(m)$
7:  Each server process $p$ in group $x_k$ executes as follows:
8:      when $x_k$-deliver$(m)$
9:          if $k = 0$ or $x_k$-delivered $m$ $(f + 1)$ times then
10:             for each $x_{k+1} \in children(x_k)$ such that
                        $m.dst \cap reach(x_{k+1}) \neq \emptyset$ do
11:                 $x_{k+1}$-broadcast$(m)$
12:         if $x_k \in m.dst$ and $m \notin A\text{-}delivered$ then
13:             a-deliver$(m)$
14:             $A\text{-}delivered \leftarrow A\text{-}delivered \cup \{m\}$
```





Partially genuine atomic multicast protocol

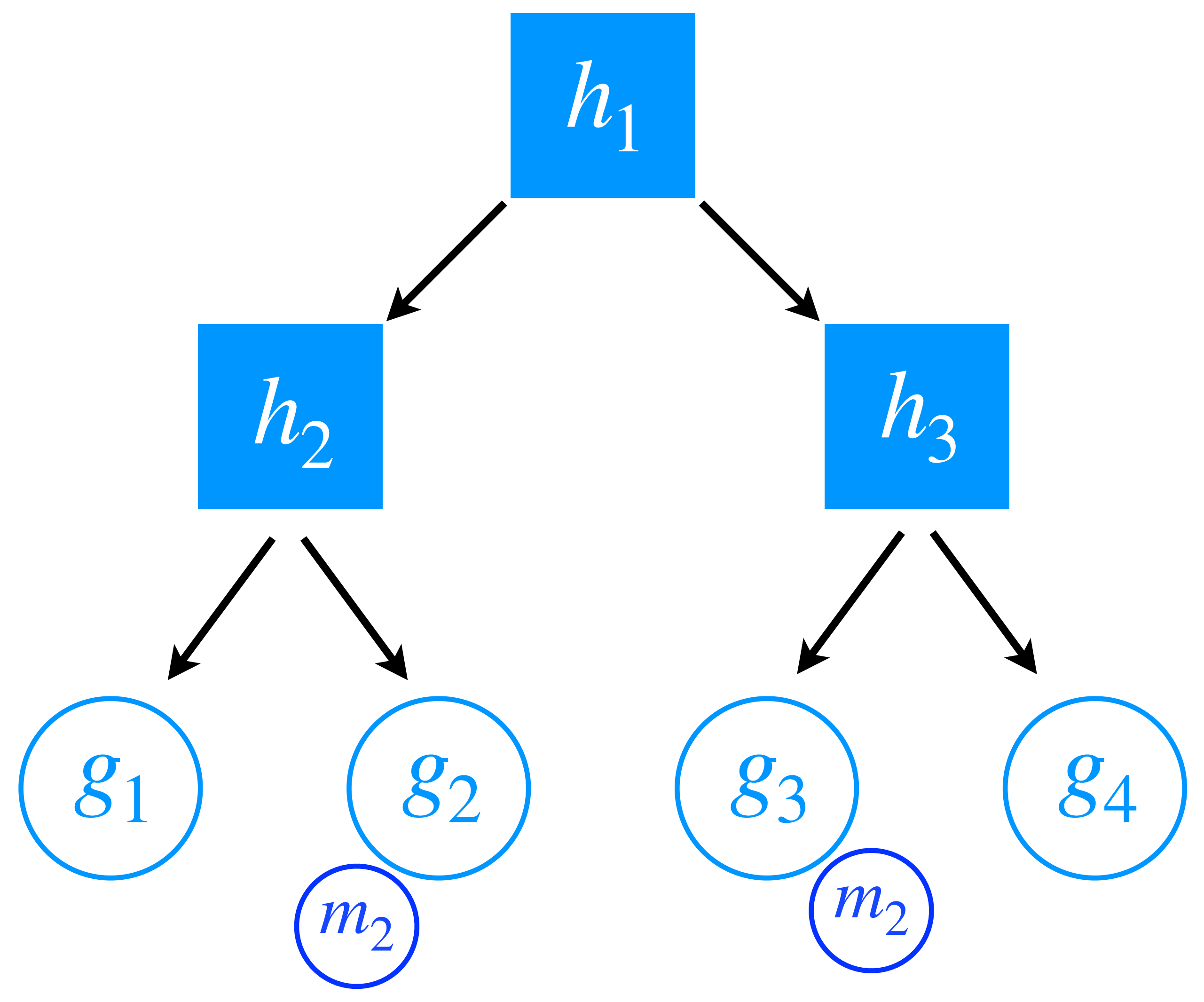
$m_2 . dst = \{g_2, g_3\}$

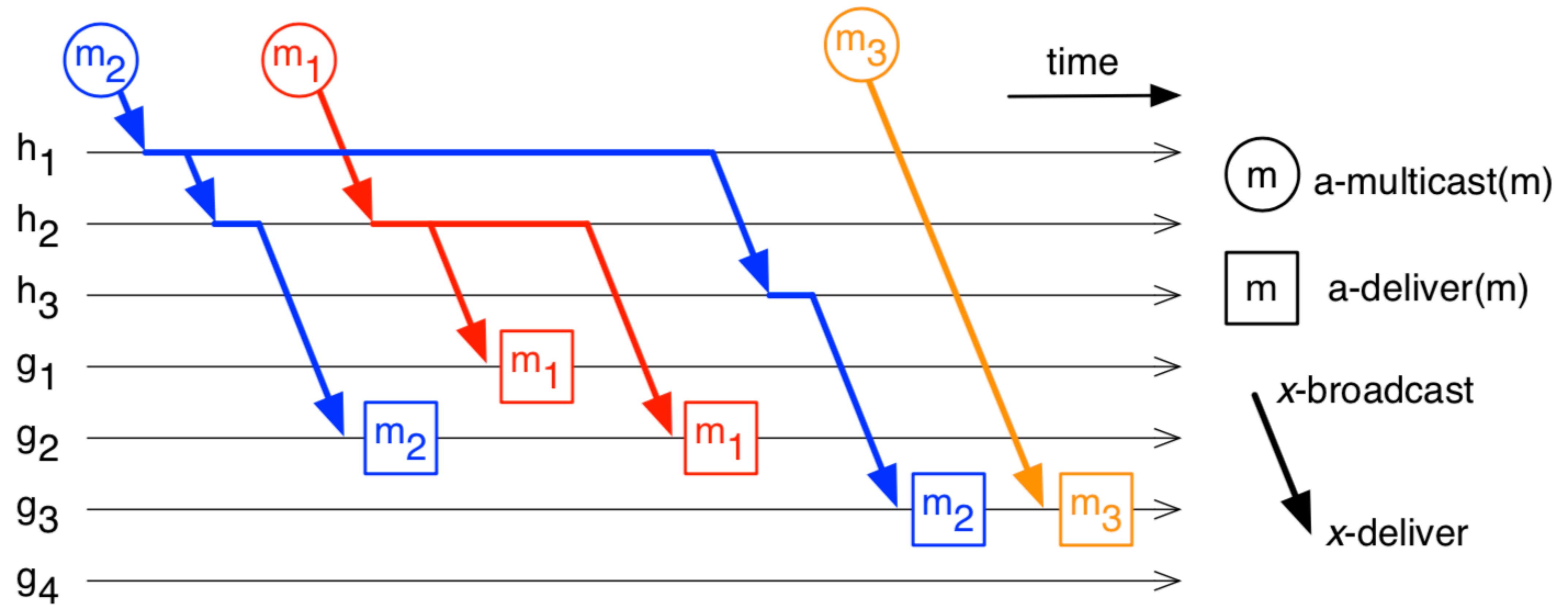
Alg. 1 ByzCast

```
1:  Initialization
2:      $\mathcal{T}$ is an overlay tree with groups $\Gamma \cup \Lambda$
3:      $A\text{-}delivered \leftarrow \emptyset$

4:  To a-multicast message $m$:
5:      $x_0 \leftarrow lca(m.dst)$          {lowest common ancestor of $m.dst$}
6:      $x_0$-broadcast($m$)

7:  Each server process $p$ in group $x_k$ executes as follows:
8:      when $x_k$-deliver($m$)
9:          if $k = 0$ or $x_k$-delivered $m$ $(f+1)$ times then
10:             for each $x_{k+1} \in children(x_k)$ such that
                        $m.dst \cap reach(x_{k+1}) \neq \emptyset$ do
11:                 $x_{k+1}$-broadcast($m$)
12:             if $x_k \in m.dst$ and $m \notin A\text{-}delivered$ then
13:                 a-deliver($m$)
14:                 $A\text{-}delivered \leftarrow A\text{-}delivered \cup \{m\}$
```





Optimisation problem of building an overlay tree

Optimisation problem:

Minimise the height of the various destination:

$$minimise \quad \sum_{d \in D} H(T,d)$$

where $H(T,d)$: height of the lowest common ancestor of groups in *d*

$$subject\ to \quad \forall(x) \ : \ L(T,d) \leq K(x)$$

where $L(T,d)$: load imposed on group *x*
$K(x)$: maximum performance in msg/s for group *x*

3.

What evidence supports the paper contributions?

3. What evidence supports the paper contributions?

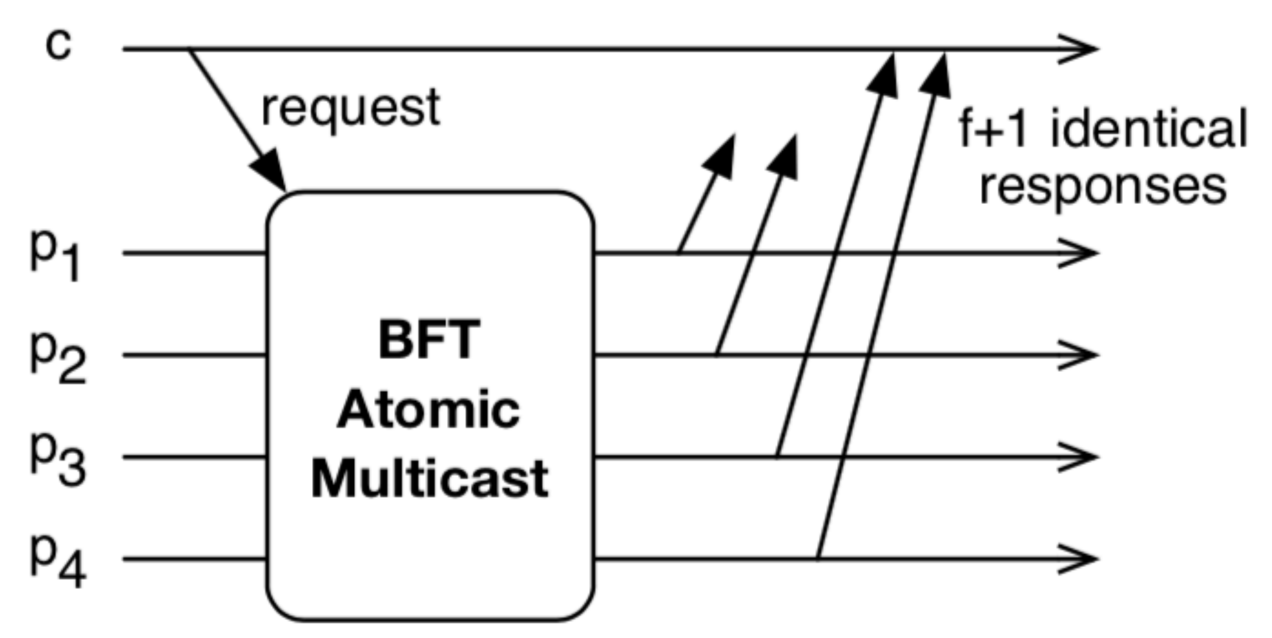
Prototype:

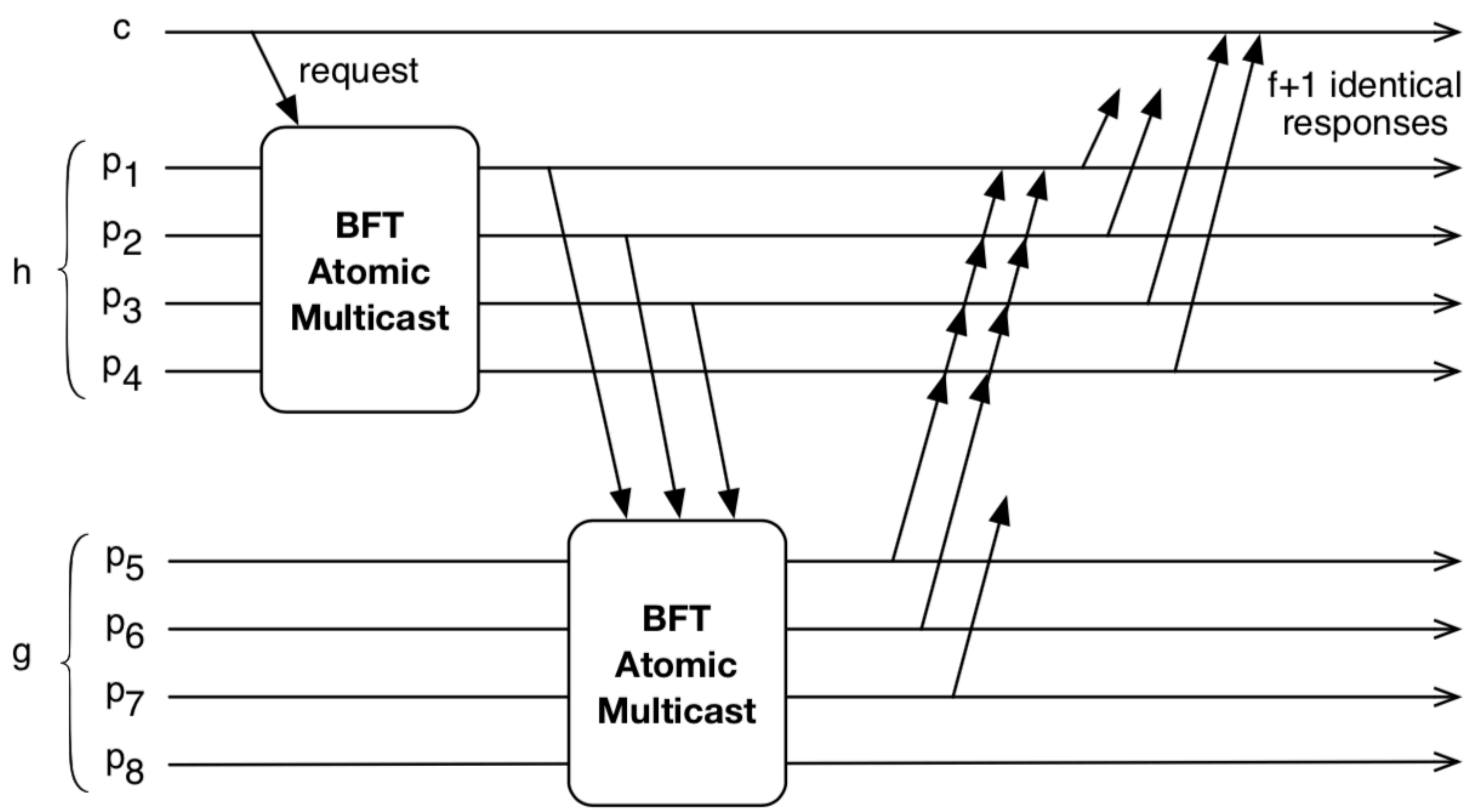
Implementation of ByzCast on top of BFT-SMaRt (Mod-SMaRt algorithm)

Each group $\longrightarrow$ BFT replicated state machine

Clients and auxiliary groups waits for $f+1$ correct replies

Implementation can be found here:
https://github.com/tarcisiocjr/byzcast





4.
Do the data actually support the conclusions?

4. Do the data actually support the conclusions?

Performance evaluation:

Environment:

Four **targets group** and up to three **auxiliary groups**

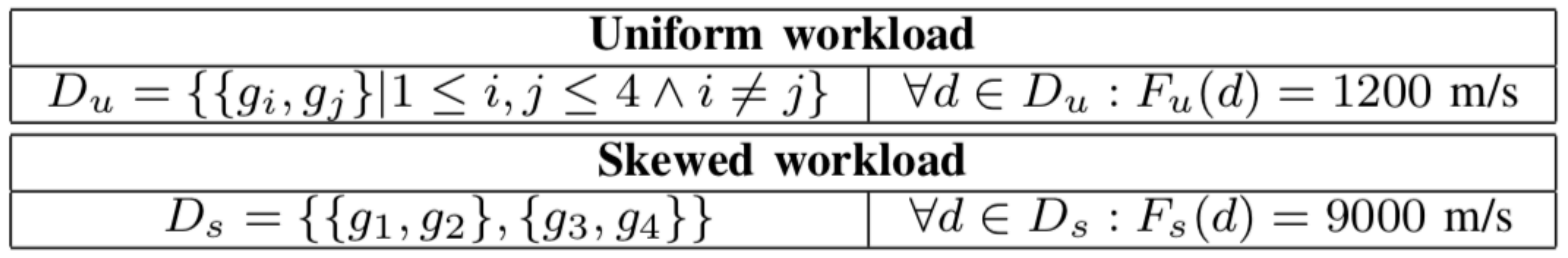
Two workloads:

uniform workload $\longrightarrow$ all combination of two destination groups

skewed workload $\longrightarrow$ destination groups are either $\{g_1, g_2\}$ or $\{g_3, g_4\}$

$$K(h_i) = 9500 \ m/s$$

4. Do the data actually support the conclusions?



Uniform workload	
$D_u = \{\{g_i, g_j\} \mid 1 \leq i, j \leq 4 \wedge i \neq j\}$	$\forall d \in D_u : F_u(d) = 1200$ m/s
Skewed workload	
$D_s = \{\{g_1, g_2\}, \{g_3, g_4\}\}$	$\forall d \in D_s : F_s(d) = 9000$ m/s

TABLE II: Uniform and skewed workloads.

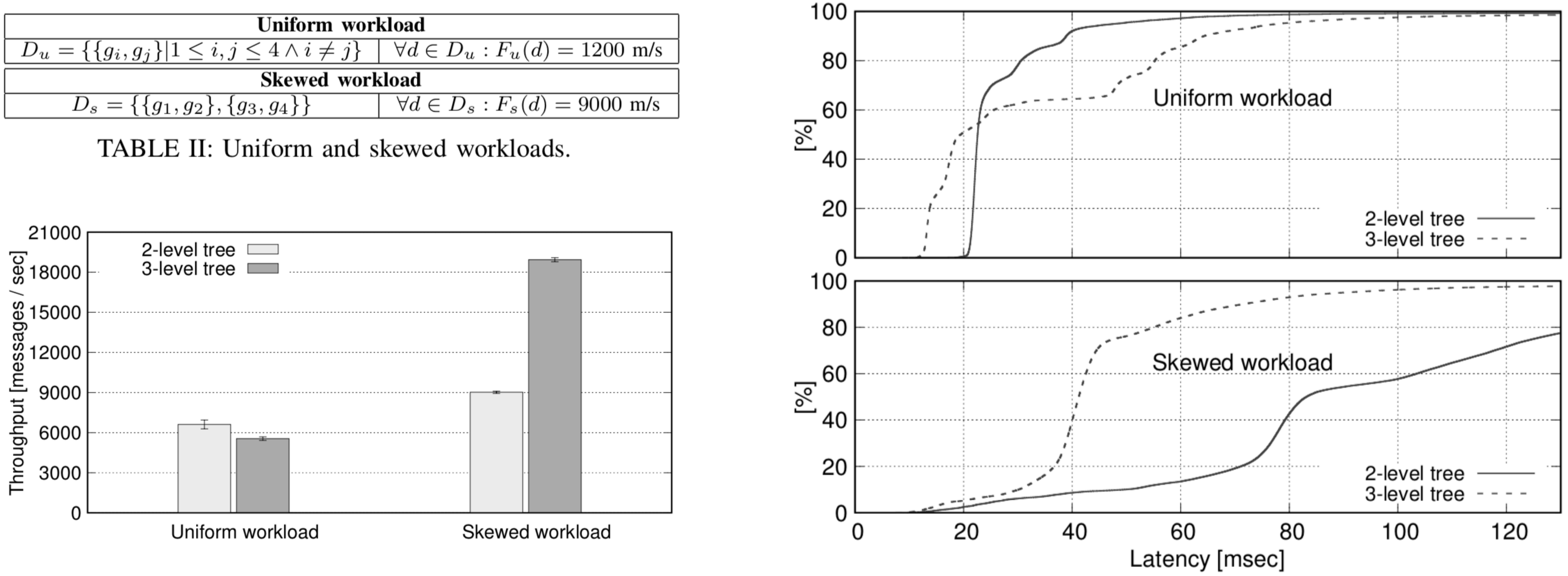
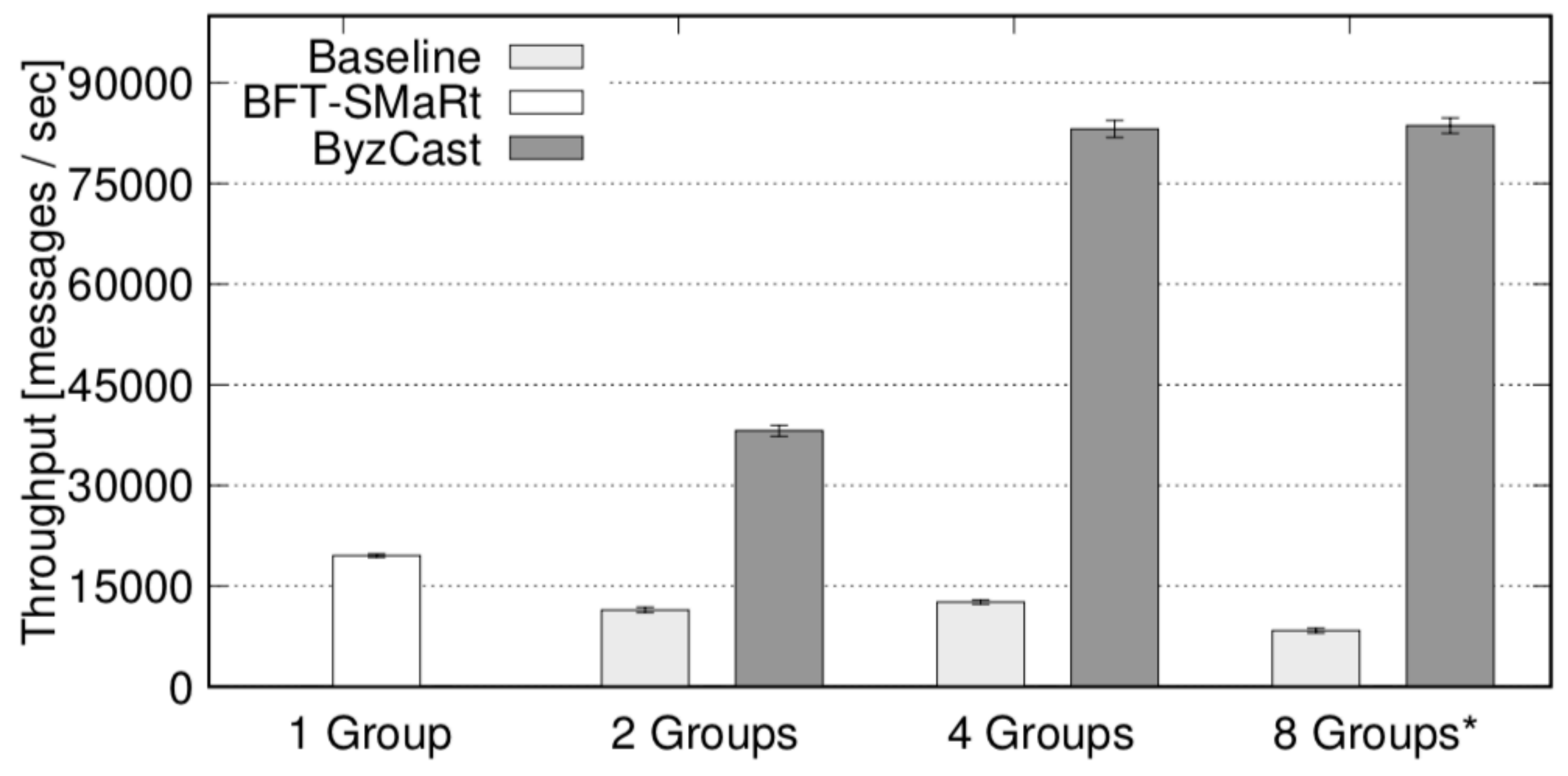


Fig. 3: ByzCast global messages throughput and latency CDF with 2-level and 3-level trees. Whiskers show 95% confidence interval.

4. Do the data actually support the conclusions?

200 clients per group multicast local messages only (100 for 8-groups)

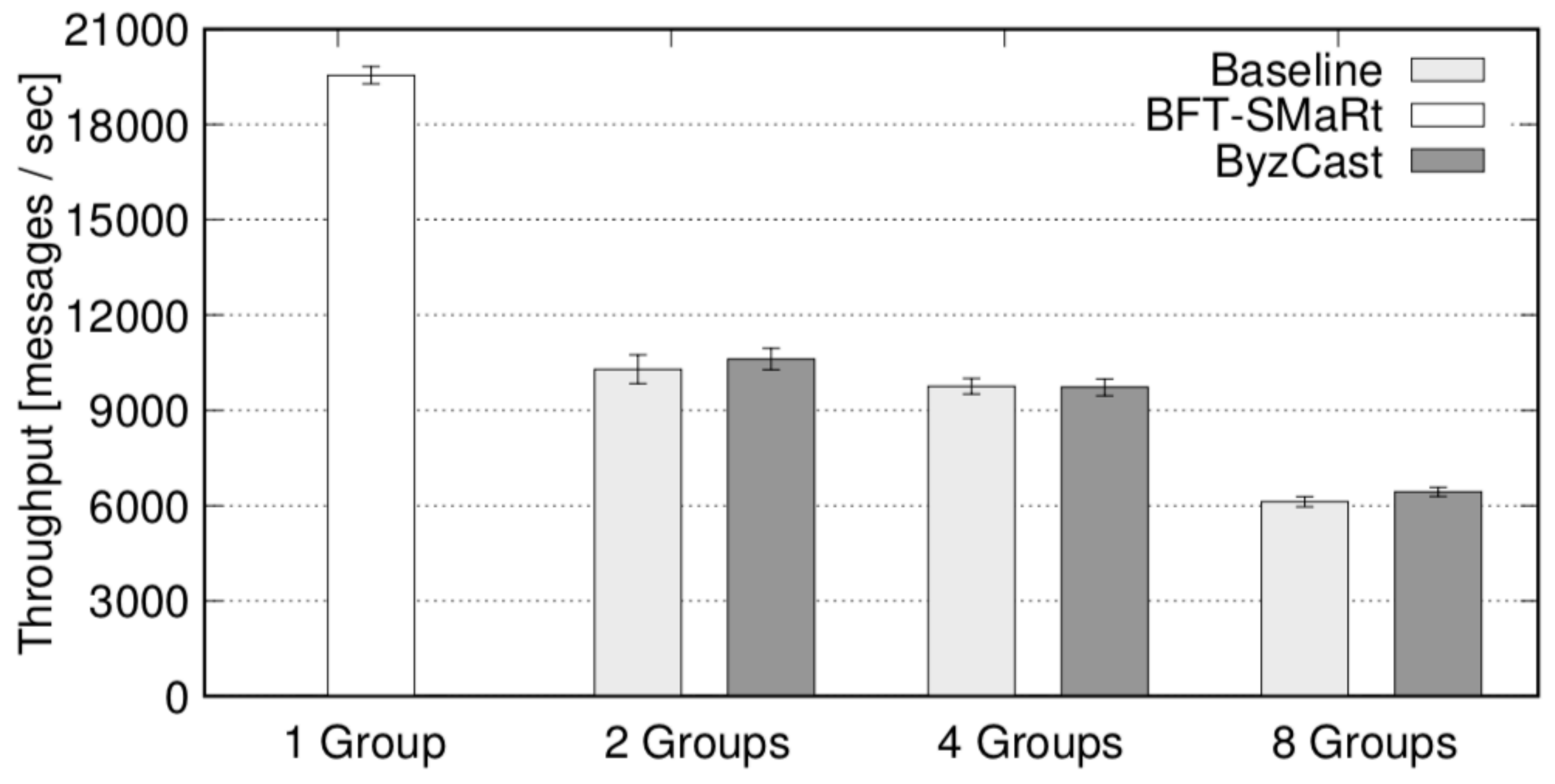


(a) Local messages.

Genuineness of ByzCast with respect to local messages leads to better performances

4. Do the data actually support the conclusions?

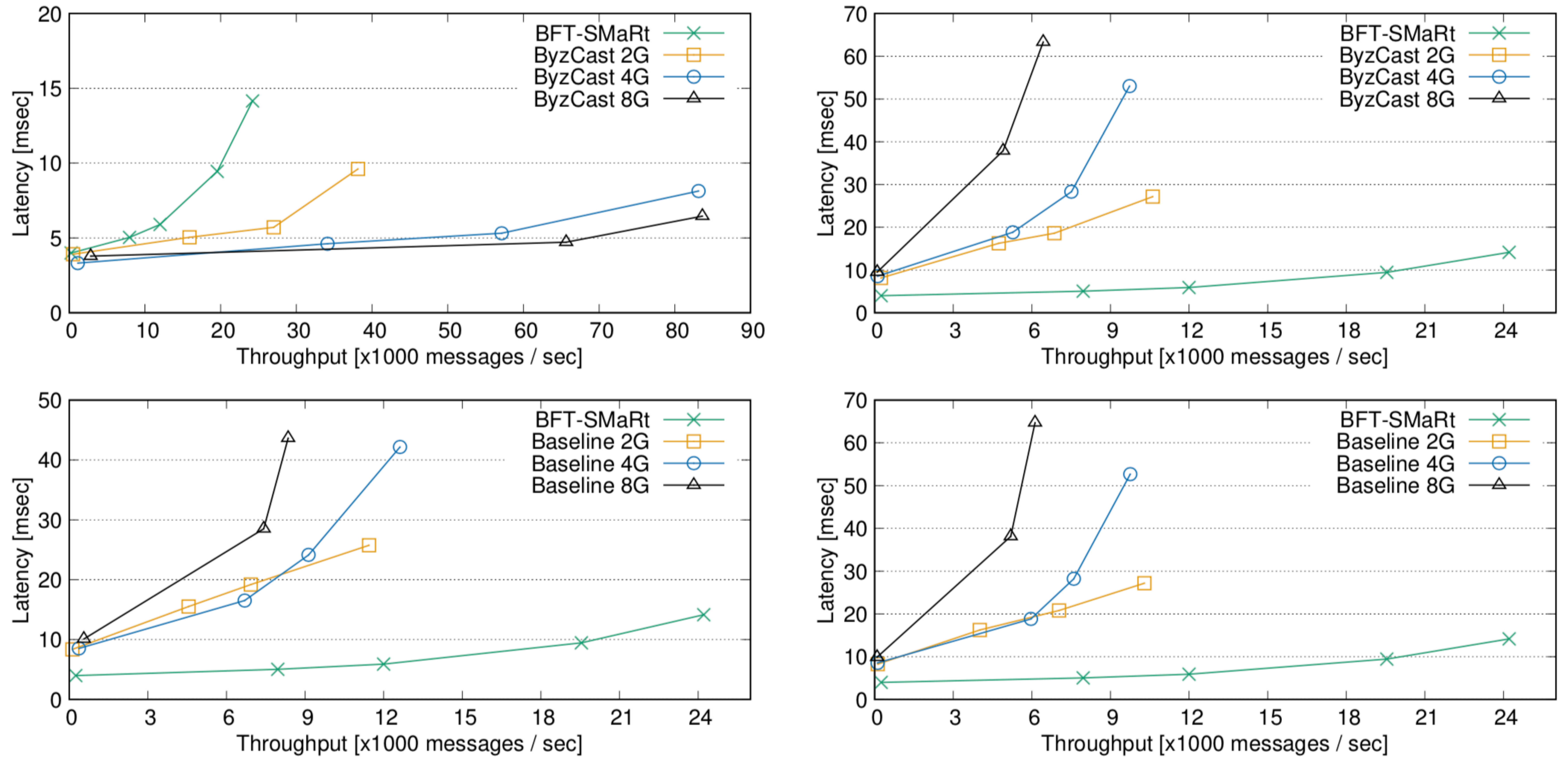
200 clients per group multicast global messages only (100 for 8-groups)



(b) Global messages.

Global messages in ByzCast have to be ordered by both the auxiliary group and the target groups

4. Do the data actually support the conclusions?



(a) Local messages: ByzCast (top) and Baseline (bottom).

(b) Global messages: ByzCast (top) and Baseline (bottom).

Fig. 5: Throughput vs. latency in a LAN.

5.
What is the quality of that evidence?

5. What is the quality of that evidence?

The experiments results seem to be plausible thanks to the **partially genuineness** of ByzCast.

Other than being performed in a **controlled environment** as the LAN, corresponding experiments have been reproduced also in **WAN**, leading to similar performances and results.

5. What is the quality of that evidence?

Performance with mixed workload in WAN $\longrightarrow$ more realistic workload

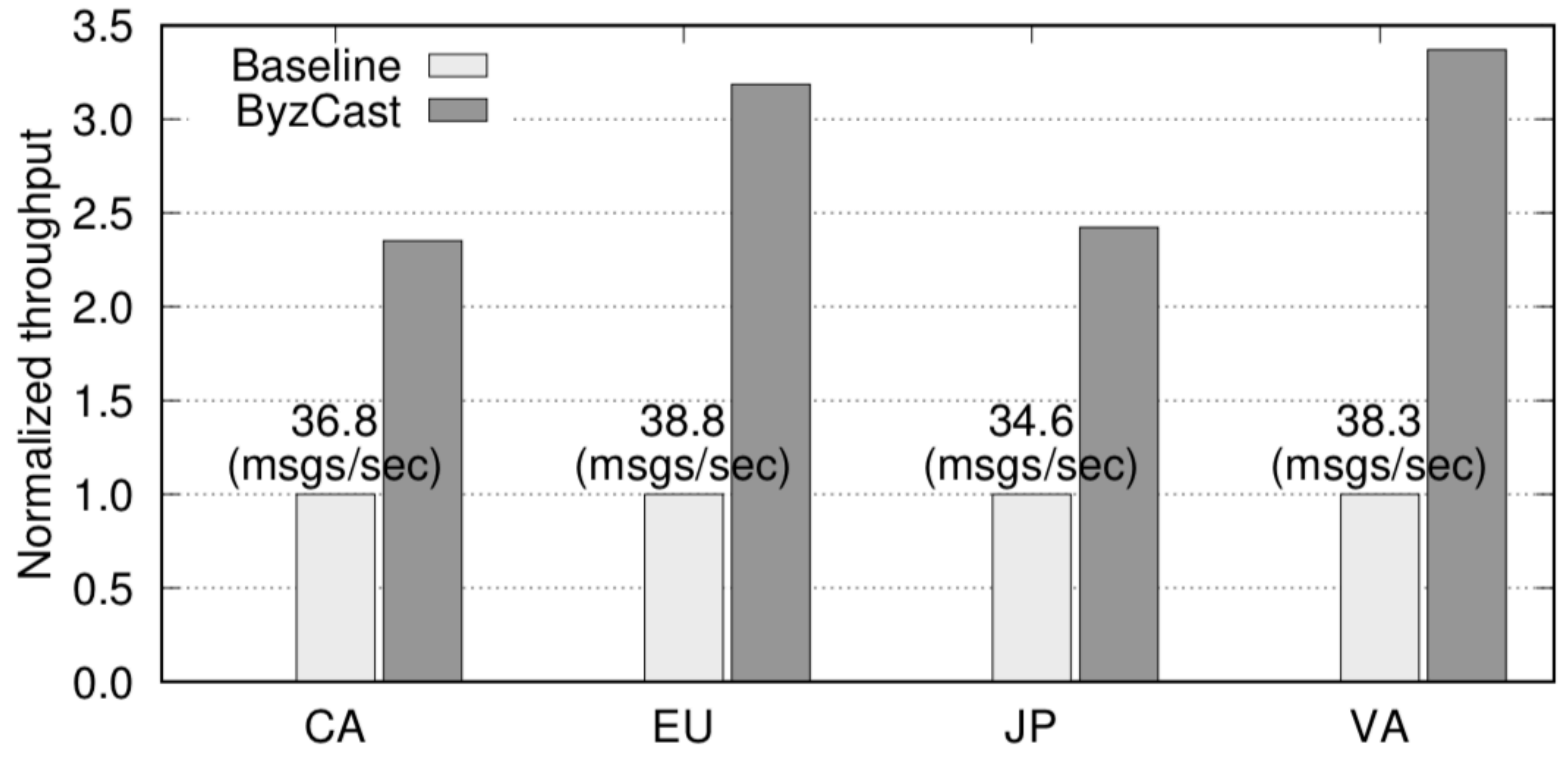


Fig. 9: Normalized throughput with mixed workload in a WAN.

ByzCast is 2x to 3x faster than the Baseline protocol in terms of throughput

6.
Why are the contributions important?

6. Why are the contributions important?

Introduction of a novel atomic multicast algorithm

Designing of a BFT and partially genuine protocol

Demonstration of performances in two different environments

Thank you for your attention

Alessia Ruggeri & Francesco Saverio Zuppichini