Scalable View-Dependent Progressive Mesh Streaming

黃瑋璨

新加坡國立大學

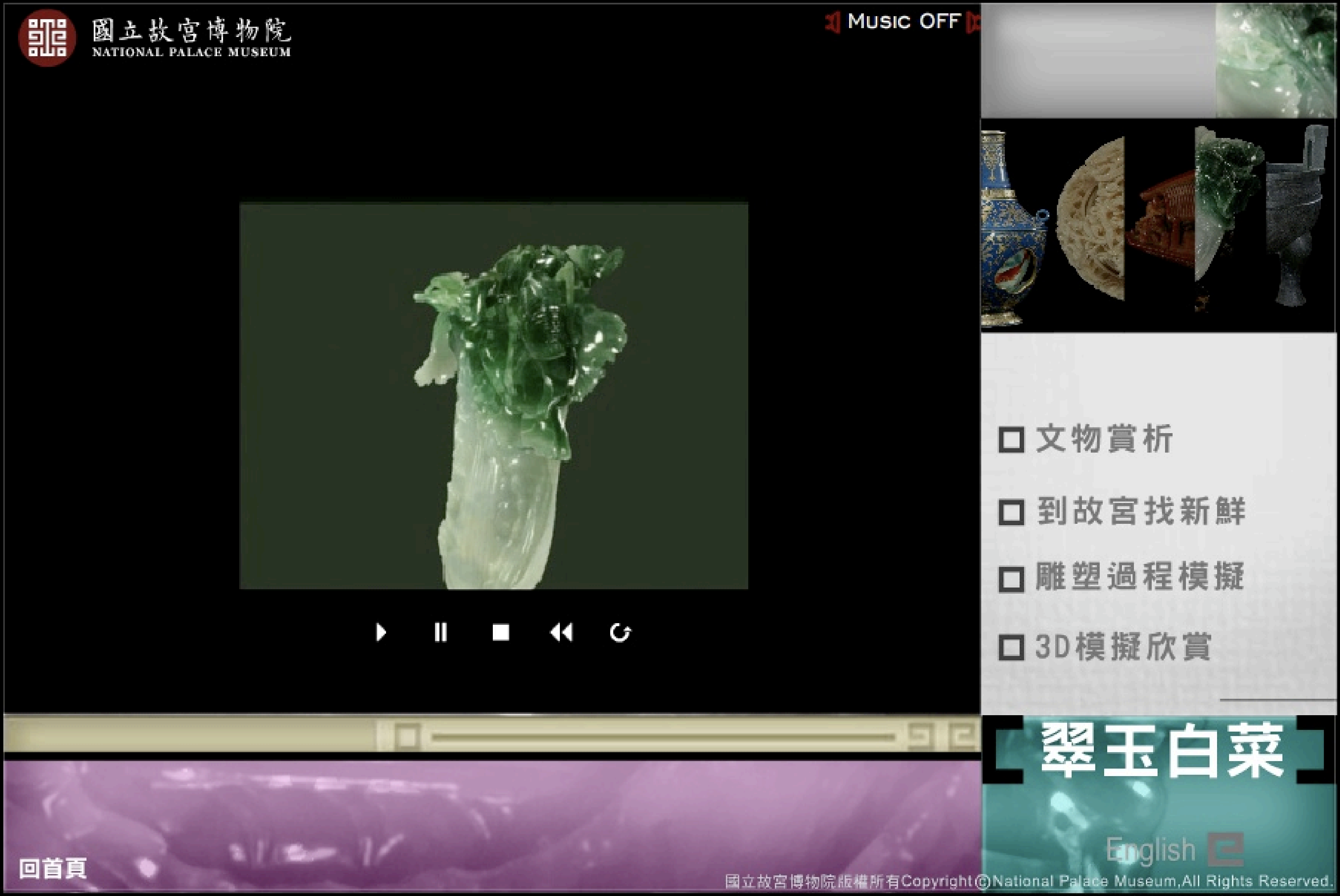
WEI TSANG **OOI**

National University of Singapore

joint work with

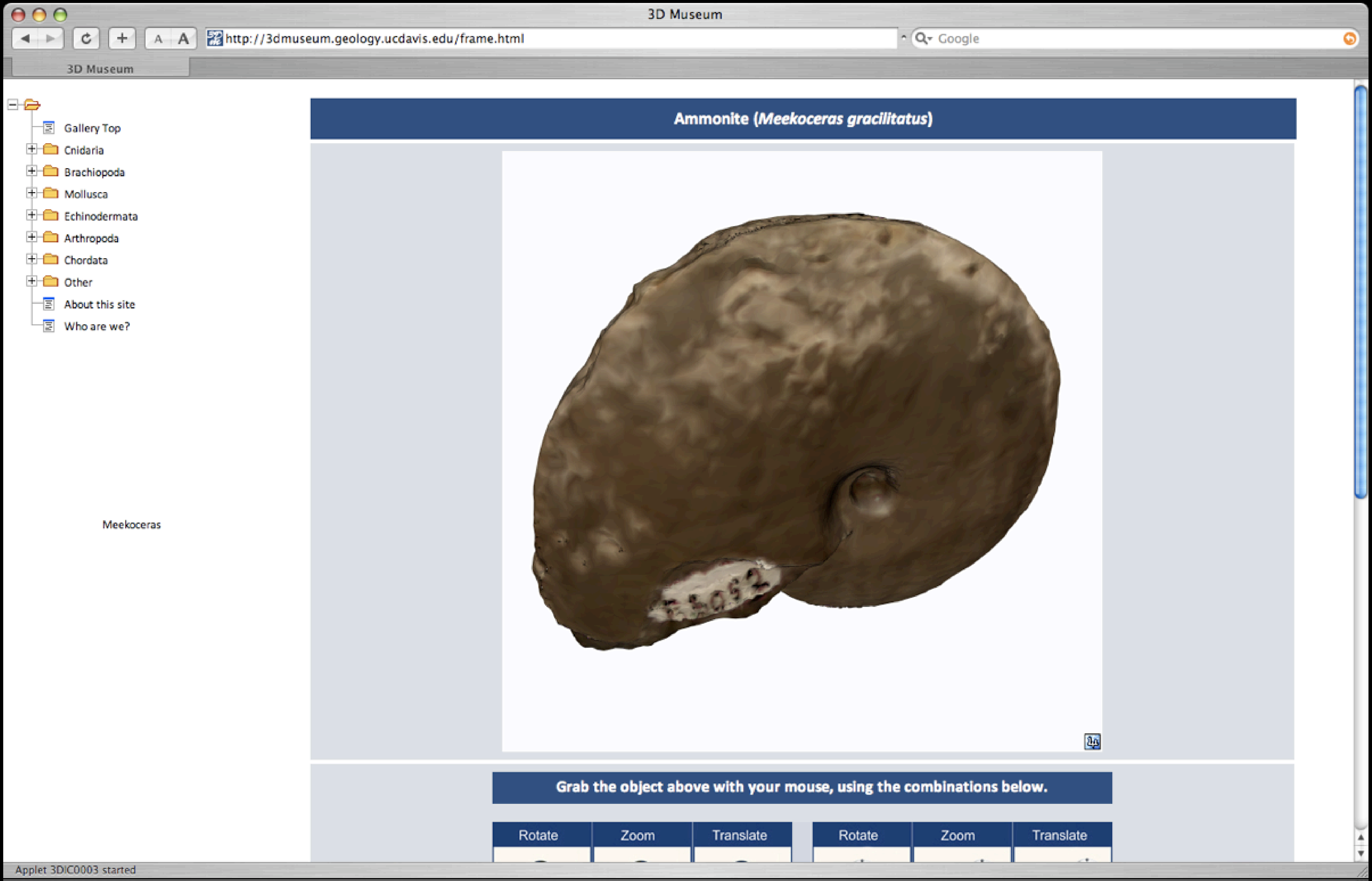
Cheng Wei

National University of Singapore



國立故宮博物院
NATIONAL PALACE MUSEUM
Music OFF
□ 文物賞析
□ 到故宮找新鮮
□ 雕塑過程模擬
□ 3D模擬欣賞
English
回首頁

翠玉白菜



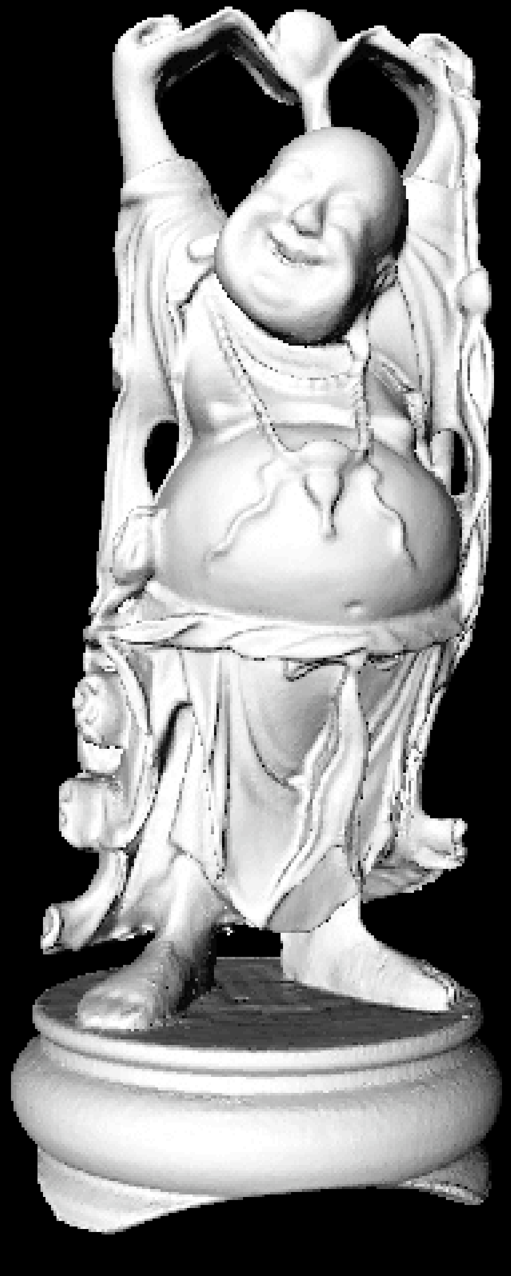
Ammonite (*Meekoceras gracilitatus*)

Grab the object above with your mouse, using the combinations below.

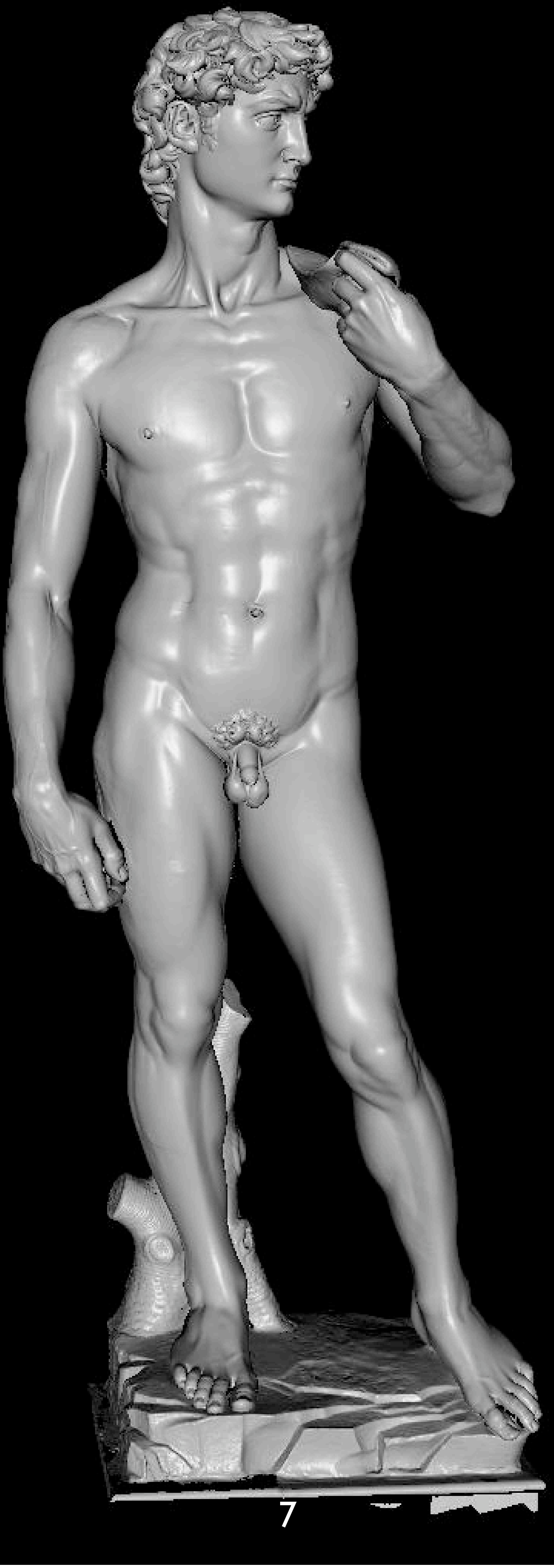
Rotate	Zoom	Translate	Rotate	Zoom	Translate

Applet 3DIC0003 started

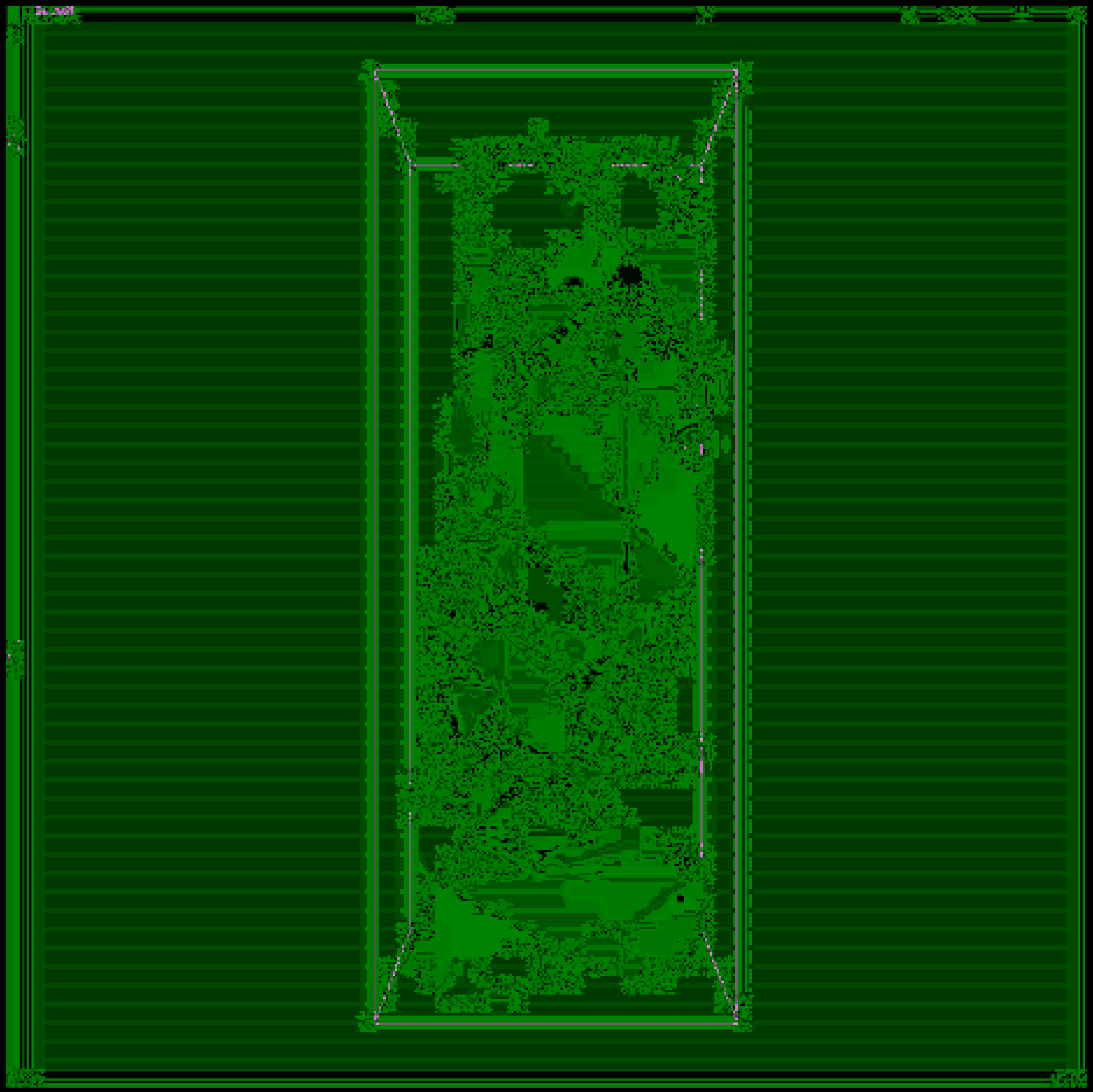
DOMINUS



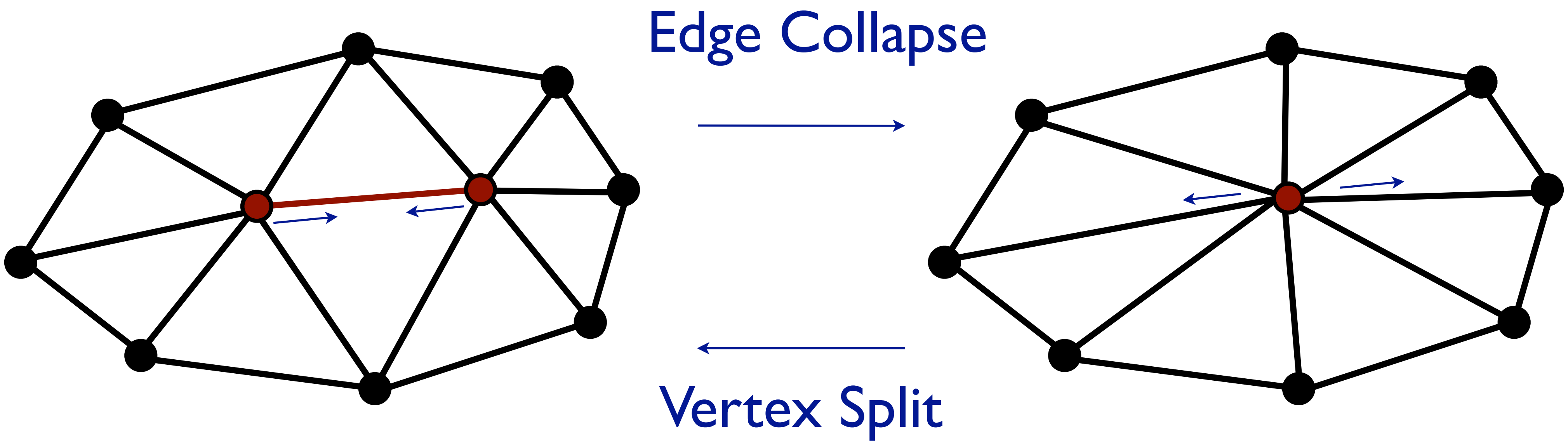
10 MB



2 GB



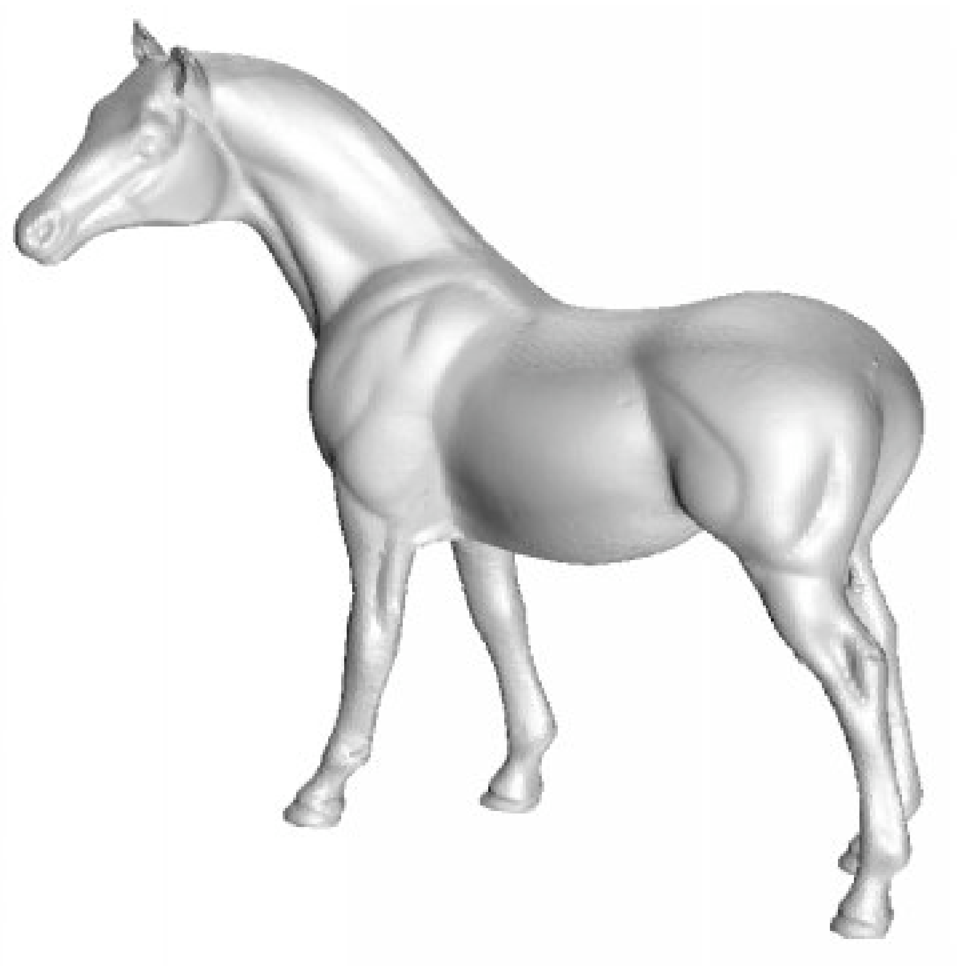
Hoppe’s Progressive Mesh



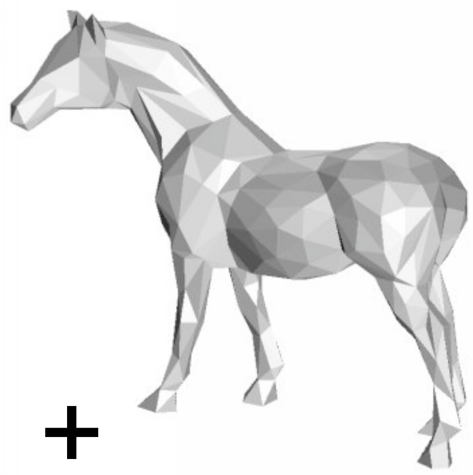
Edge Collapse

Vertex Split

At the sender



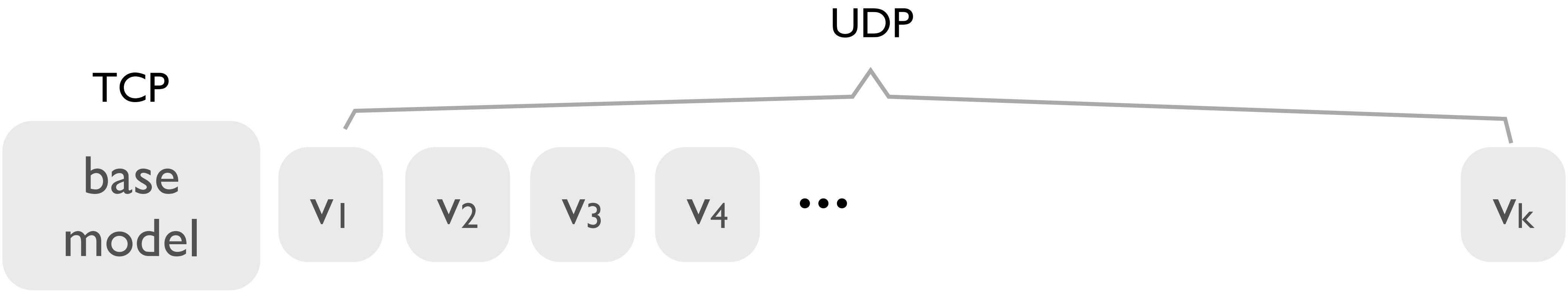
=



v_k … v_4 v_3 v_2 v_1 +

base model

Transmission



TCP

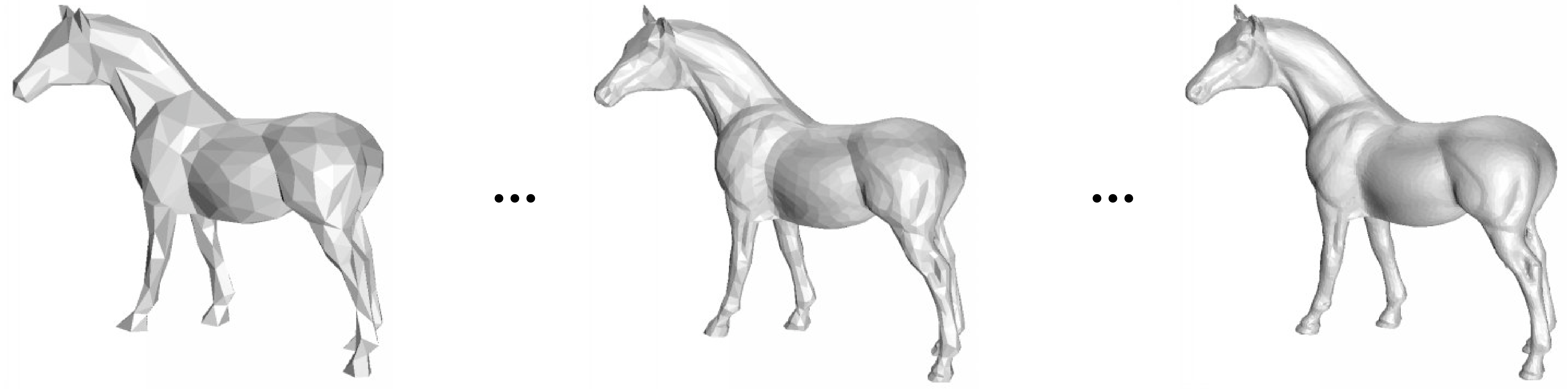
base model

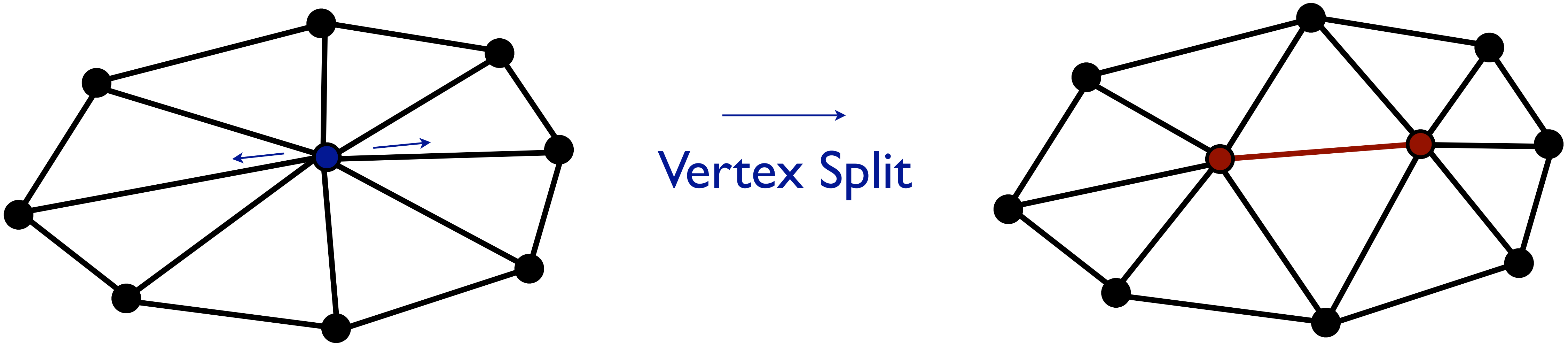
UDP

v_1 v_2 v_3 v_4 … v_k

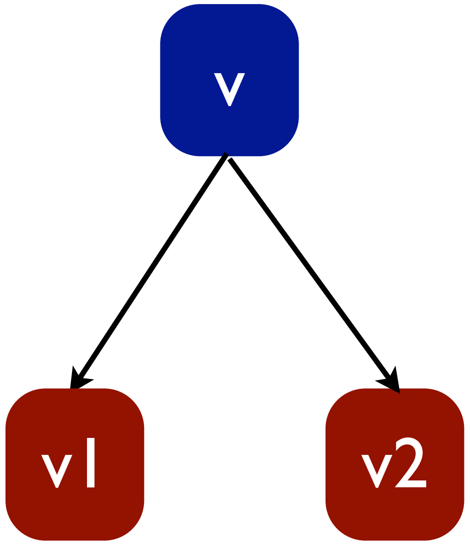
At the receiver

base model v_1 v_2 v_3 v_4 … v_k





Vertex Split

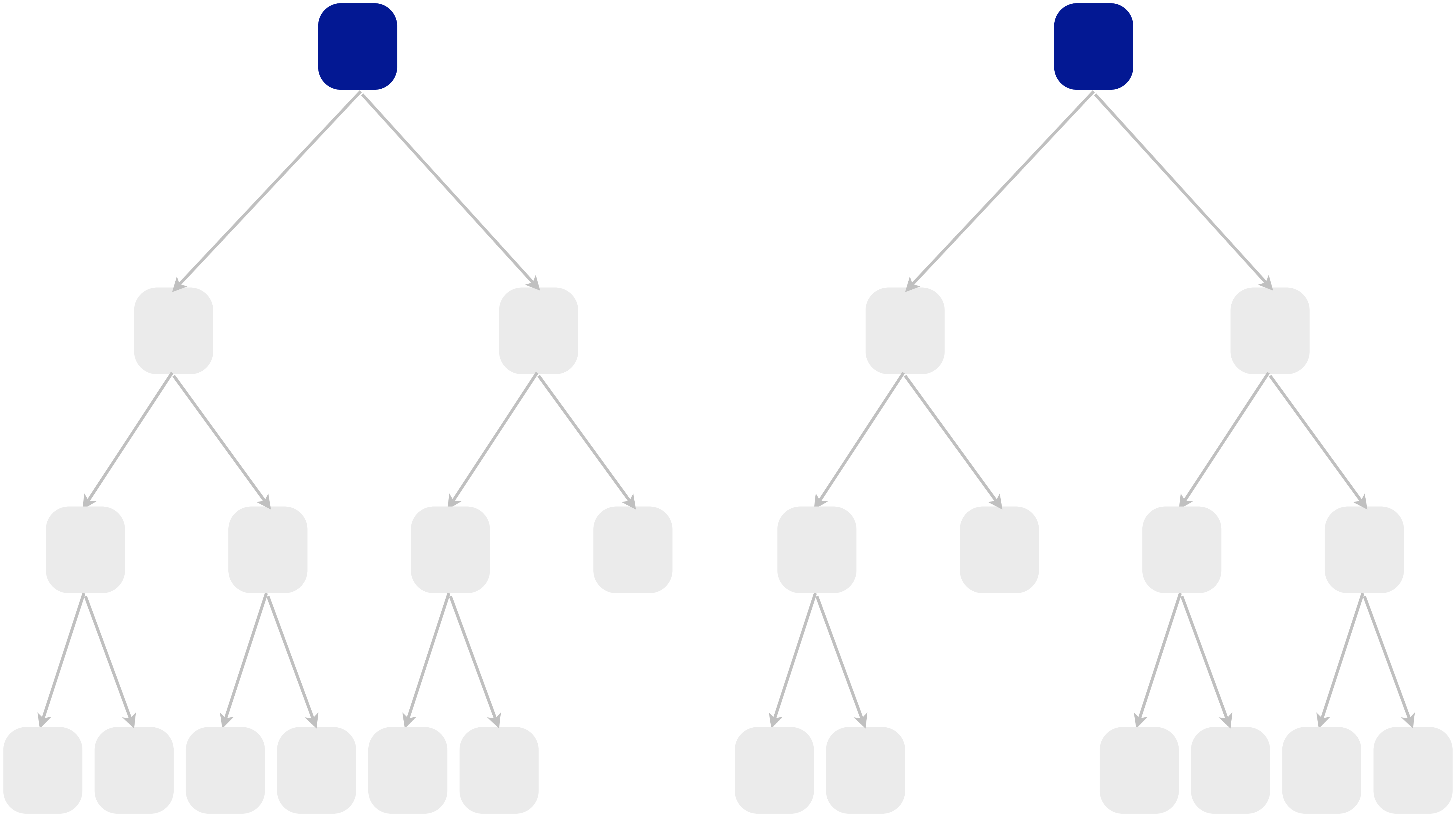


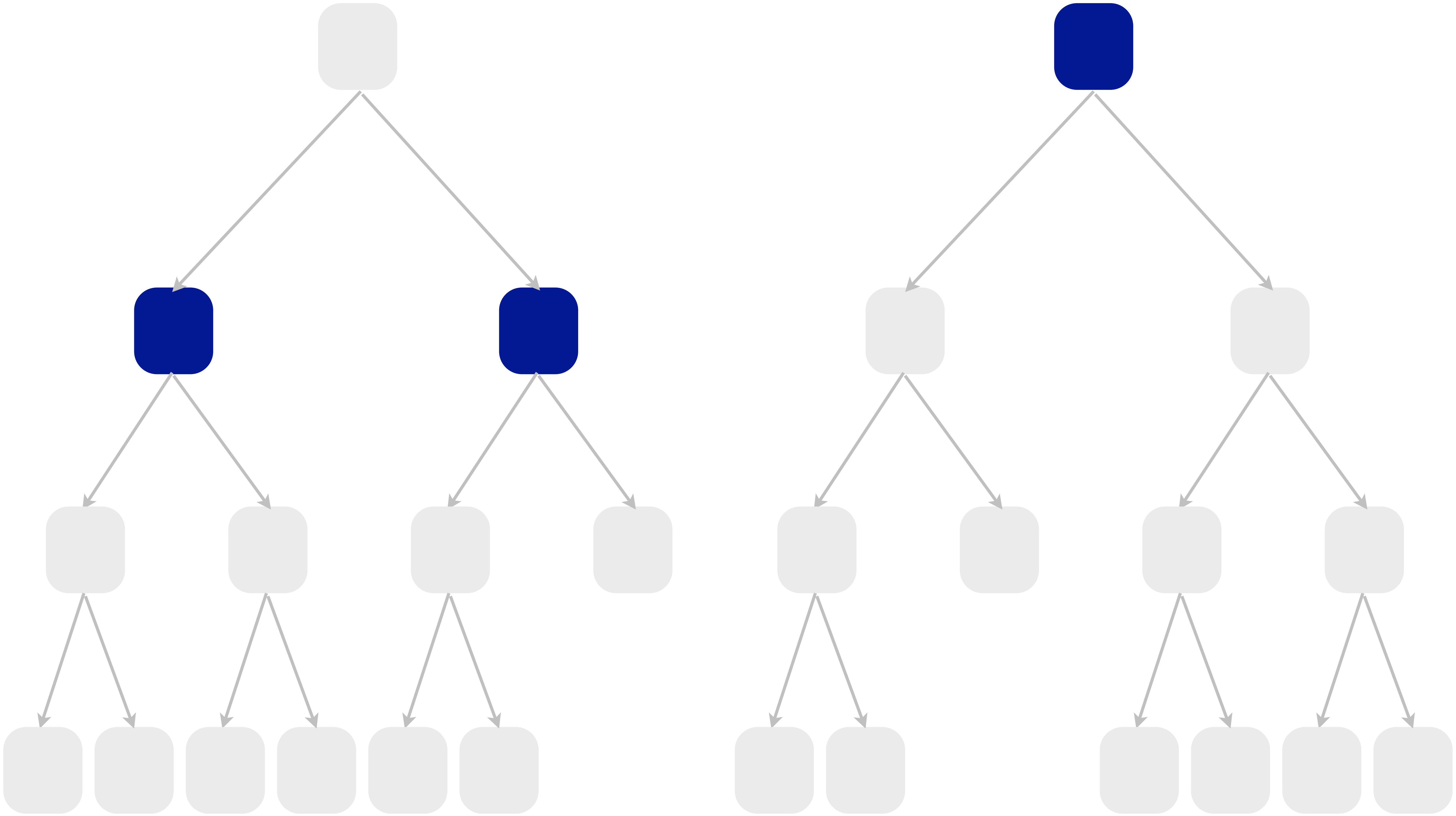
v

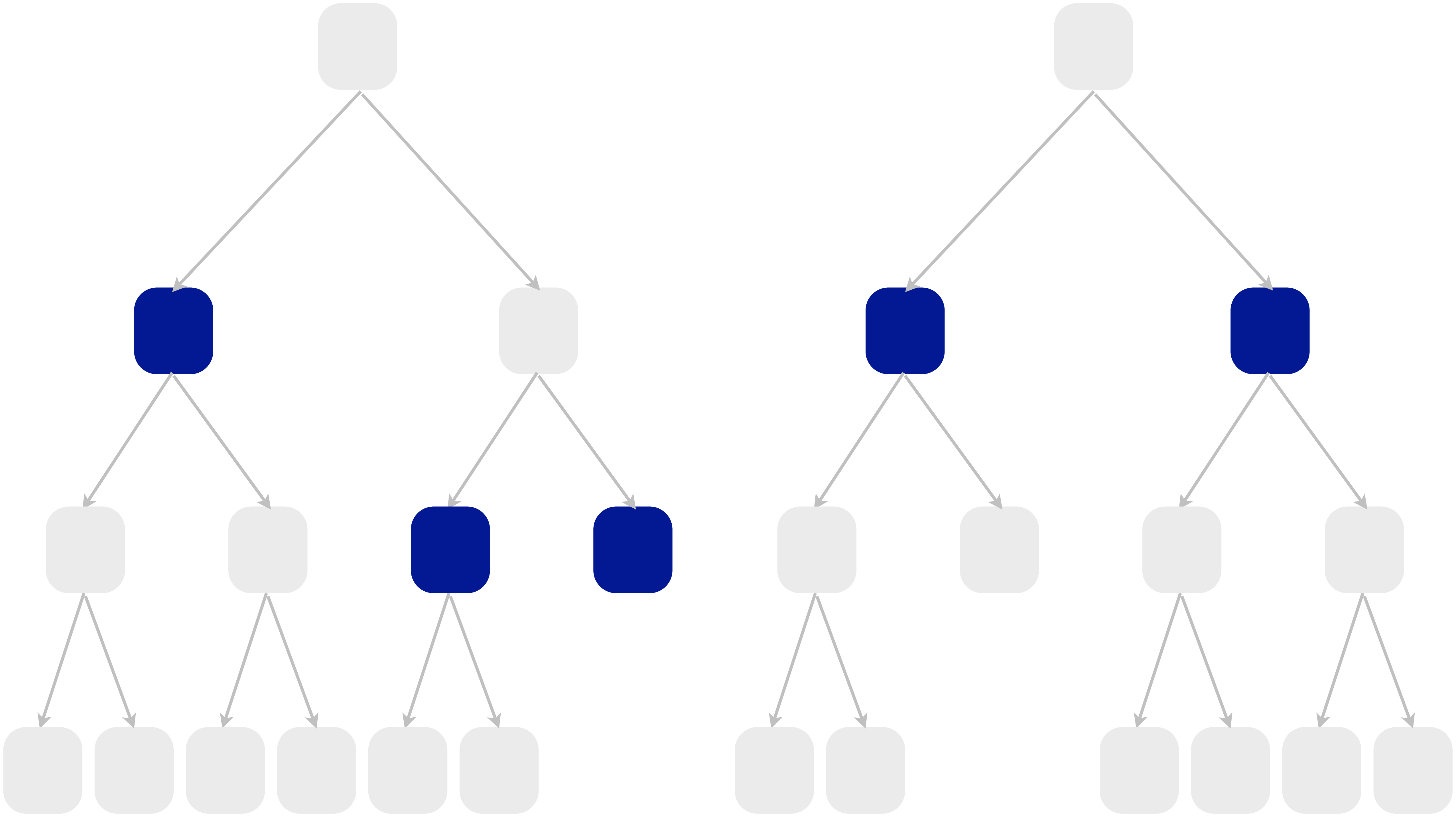
v1

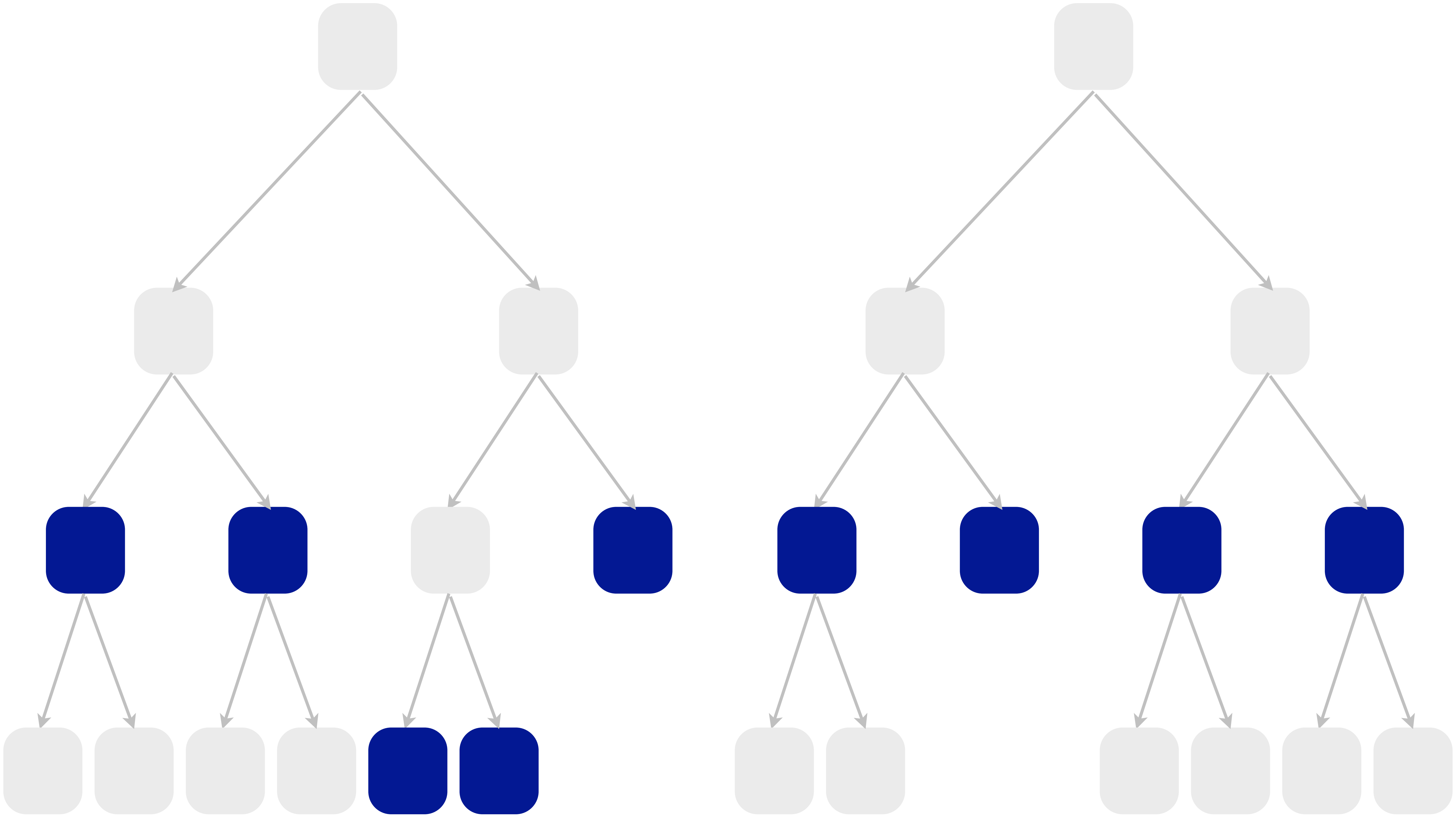
v2

base mesh

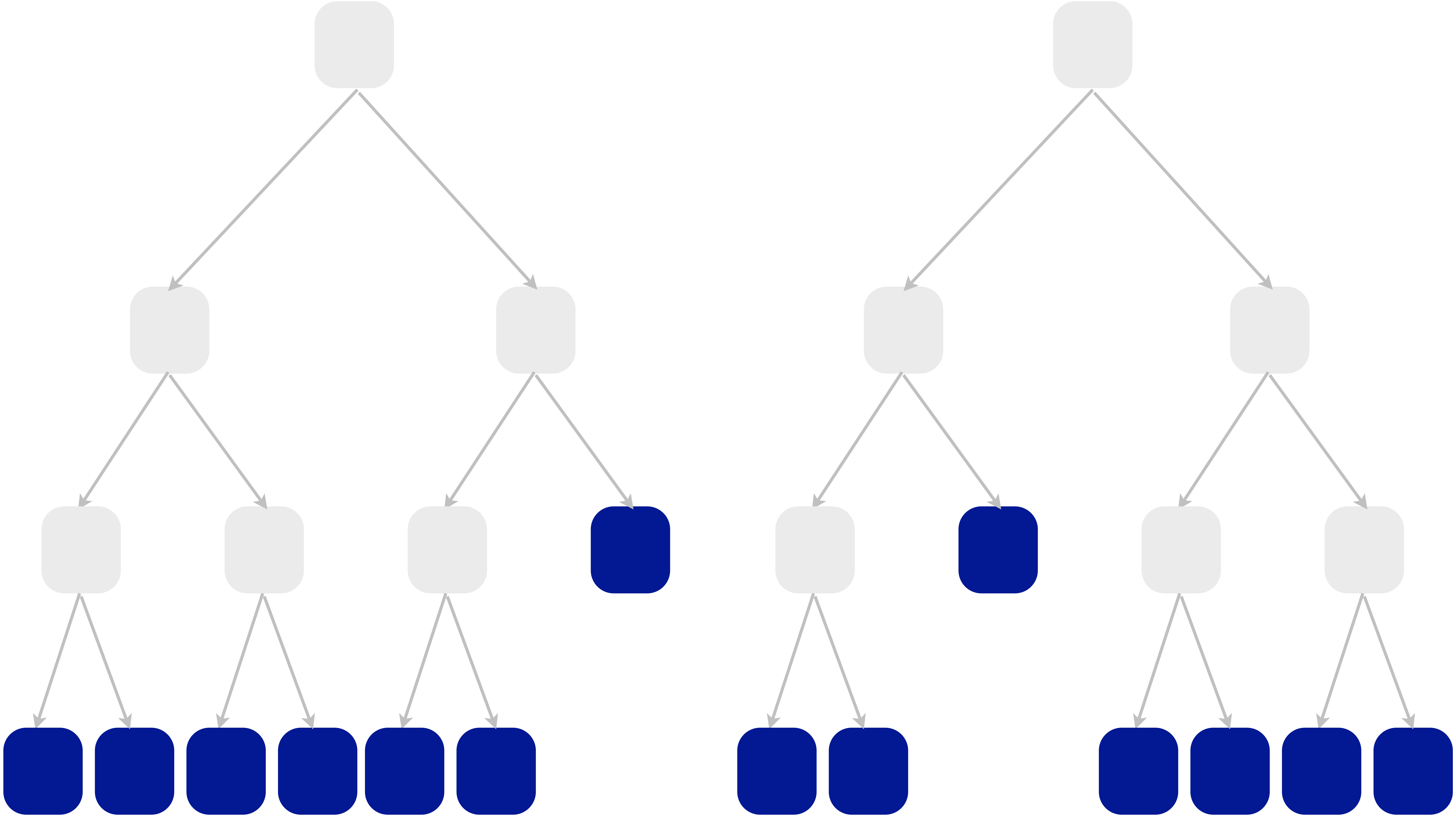






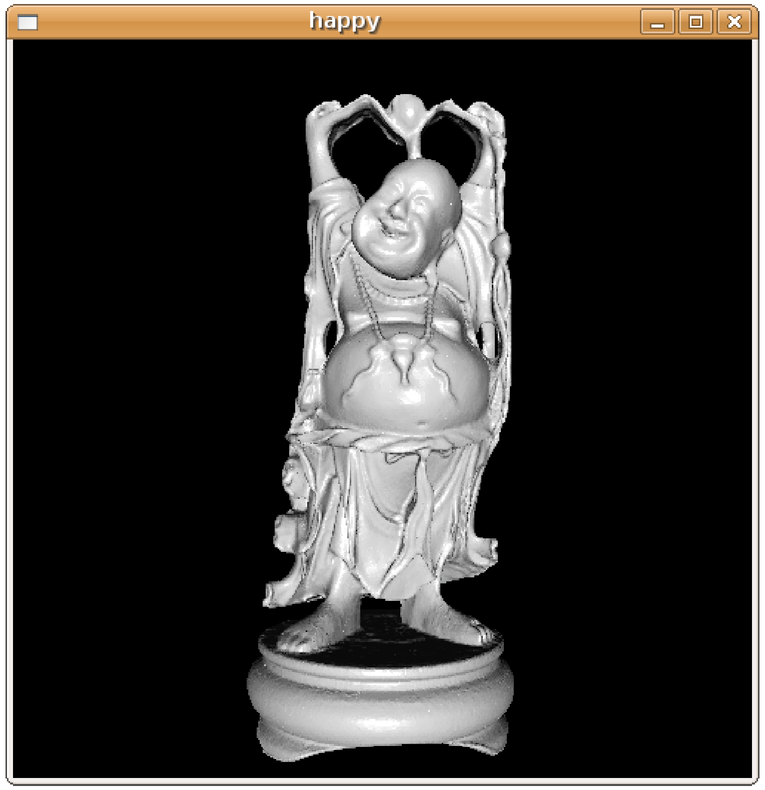


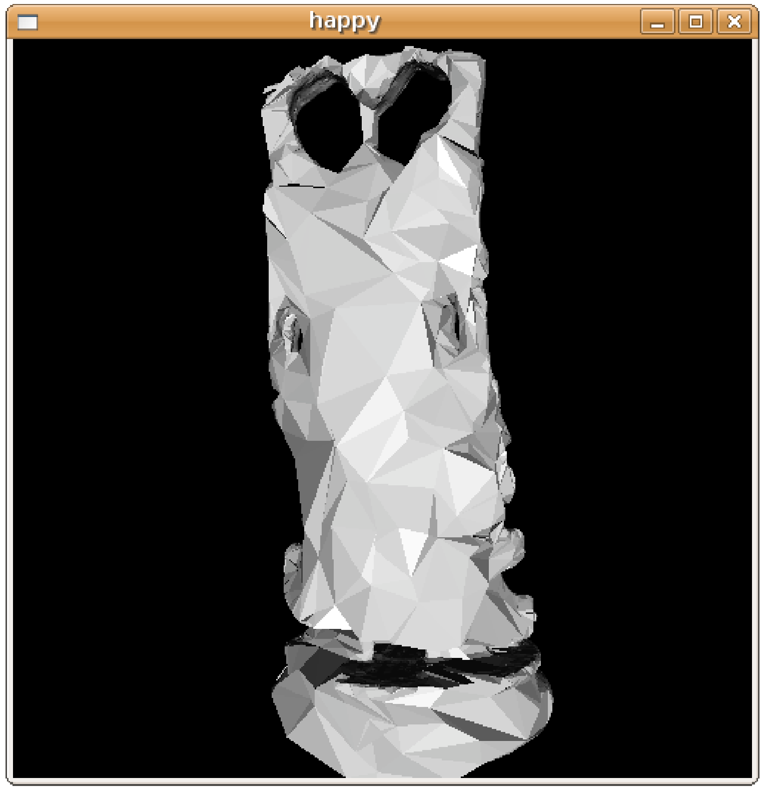
complete mesh

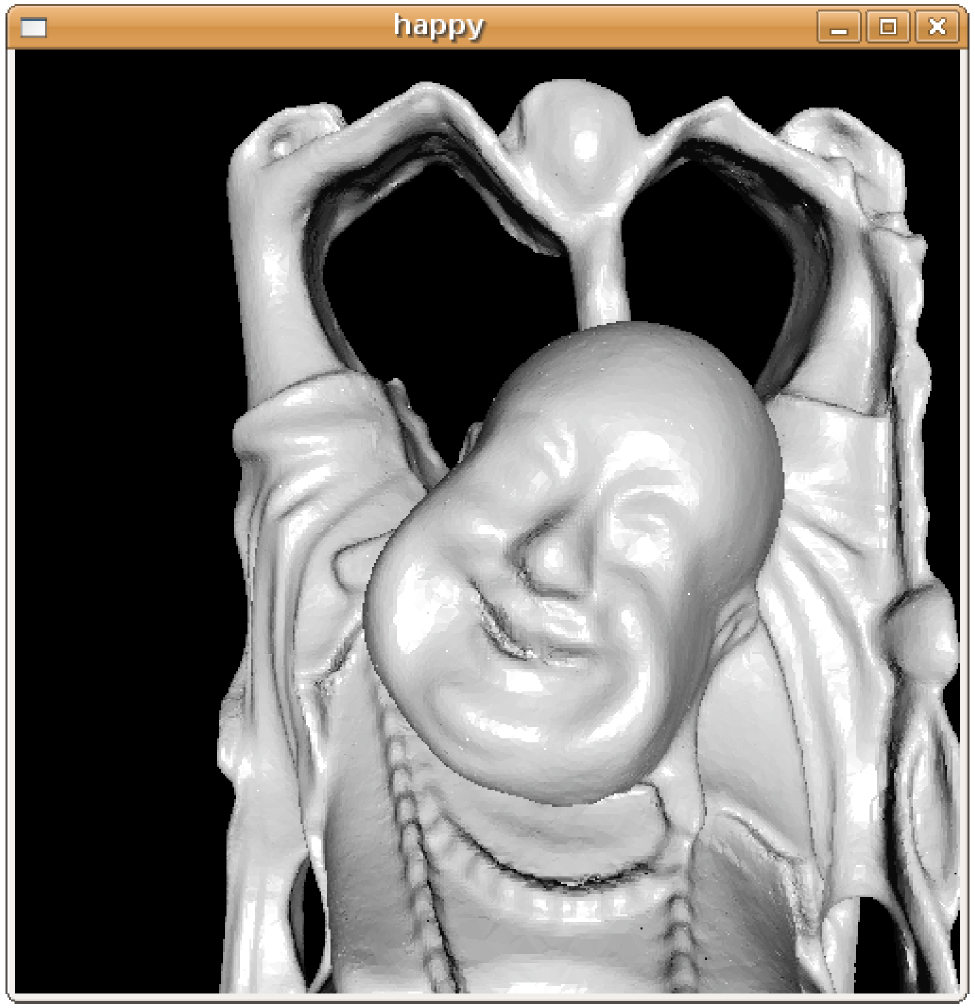


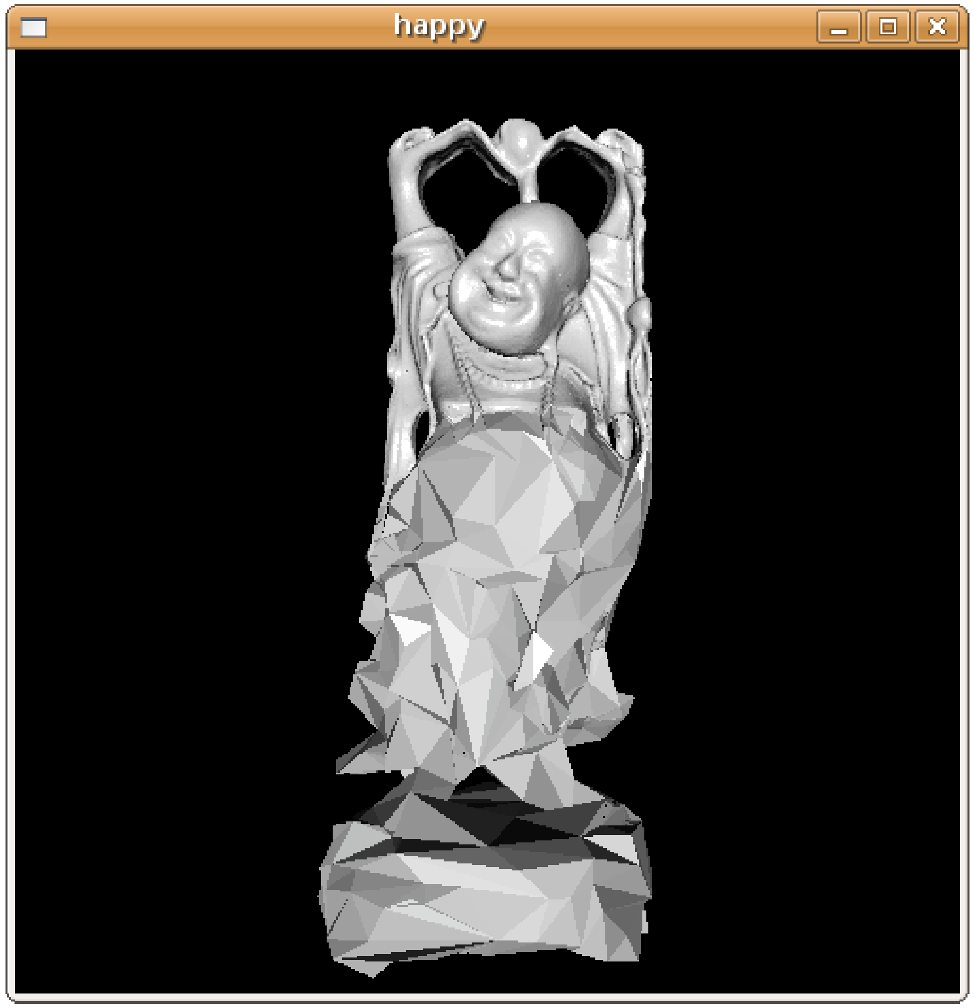
view-dependent streaming:

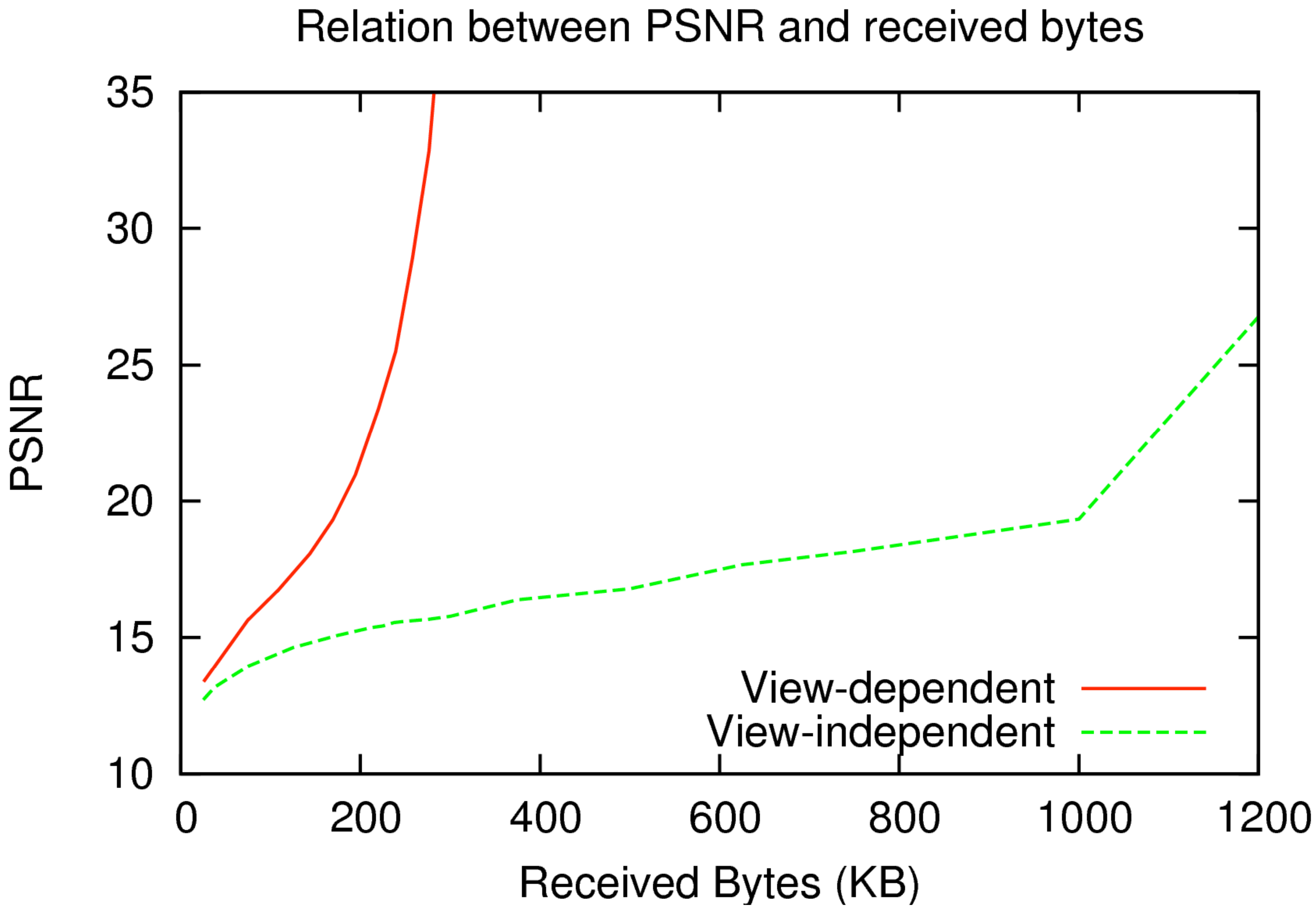
only send what the receiver can see






happy


happy


Relation between PSNR and received bytes
PSNR
35
30
25
20
15
10
0
200
400
600
800
1000
1200
Received Bytes (KB)
View-dependent
View-independent

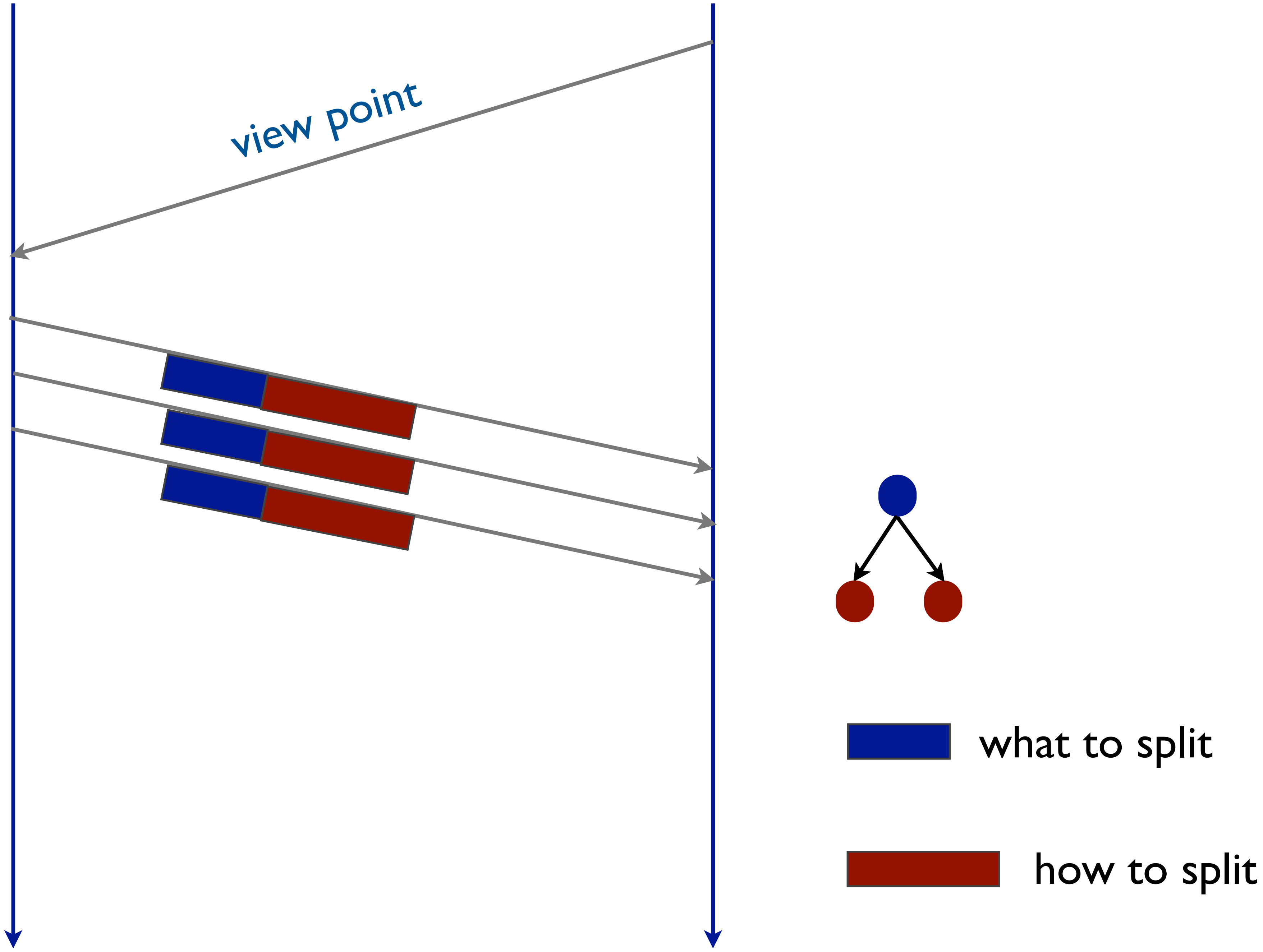
what to send?

in what order?

what to send?
determined by view point

in what order?
determined by visual contributions

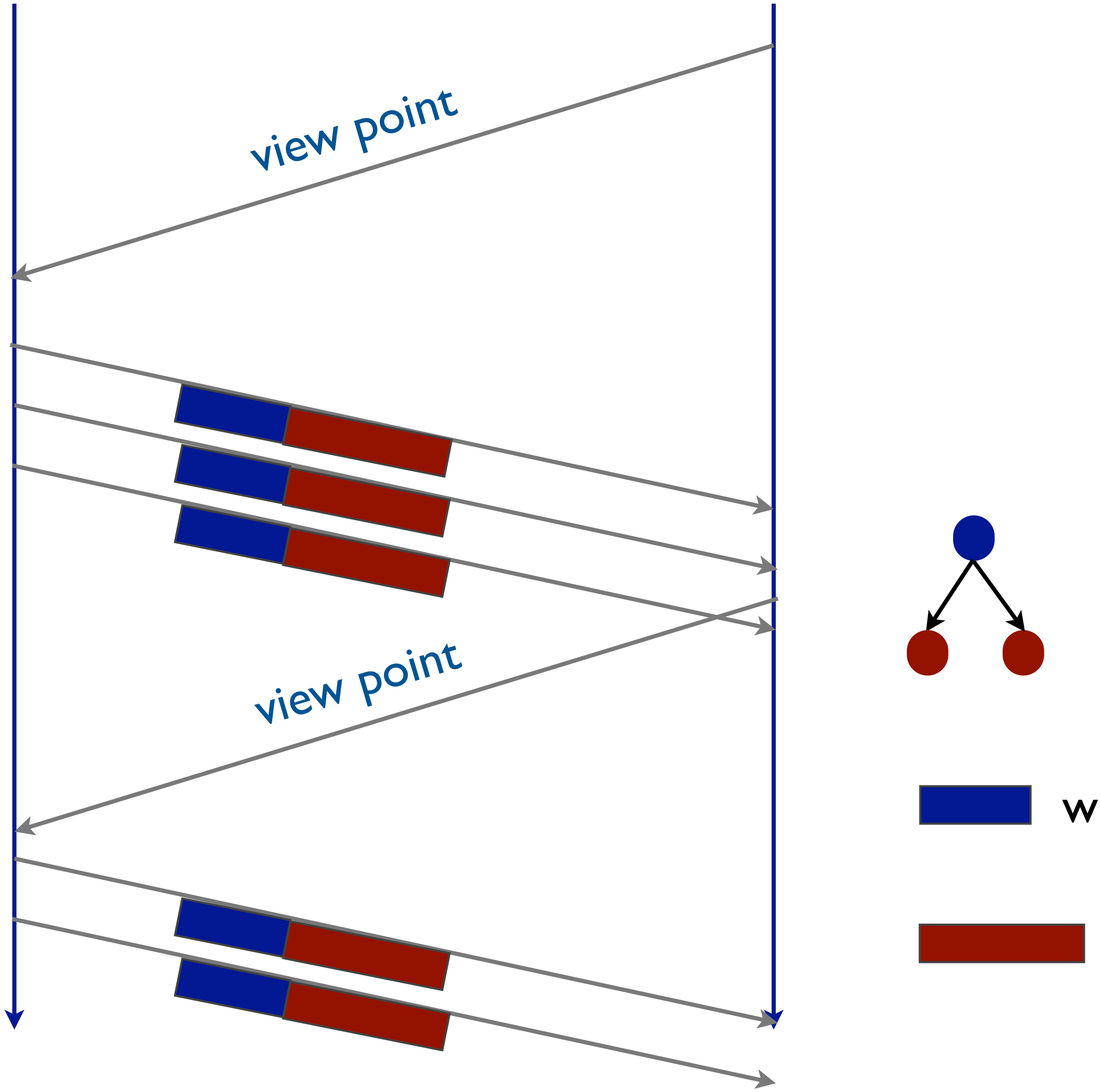
Existing Approach



🟦 what to split

🟥 how to split

Existing Approach



🟦 what to split

🟥 how to split

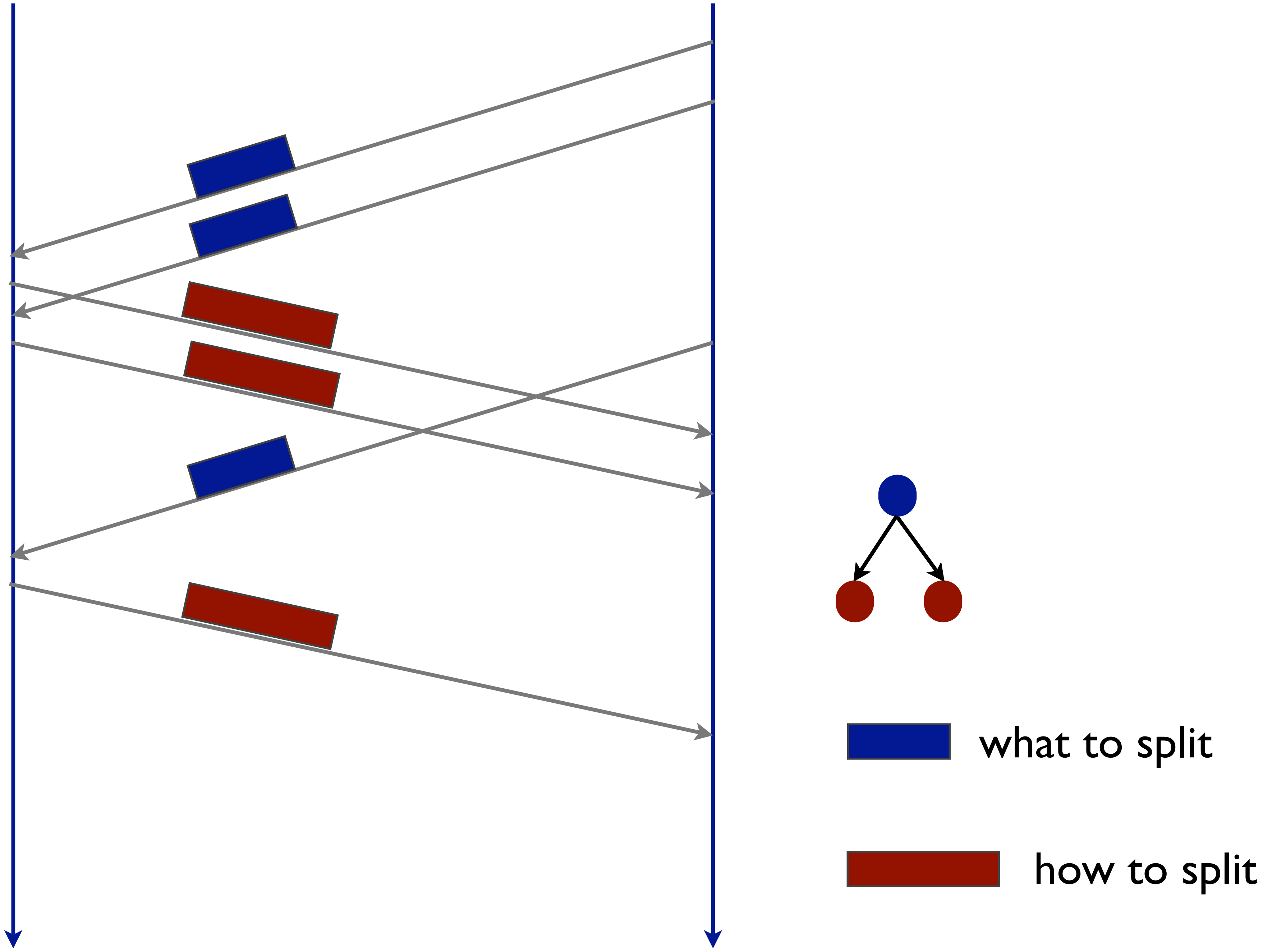
For each receiver, server needs to:

- compute visibility
- compute visual contribution of each vertex split
- sort vertex splits
- remember what has been sent

“dumb client, smart server”

does not scale

Receiver-driven Approach

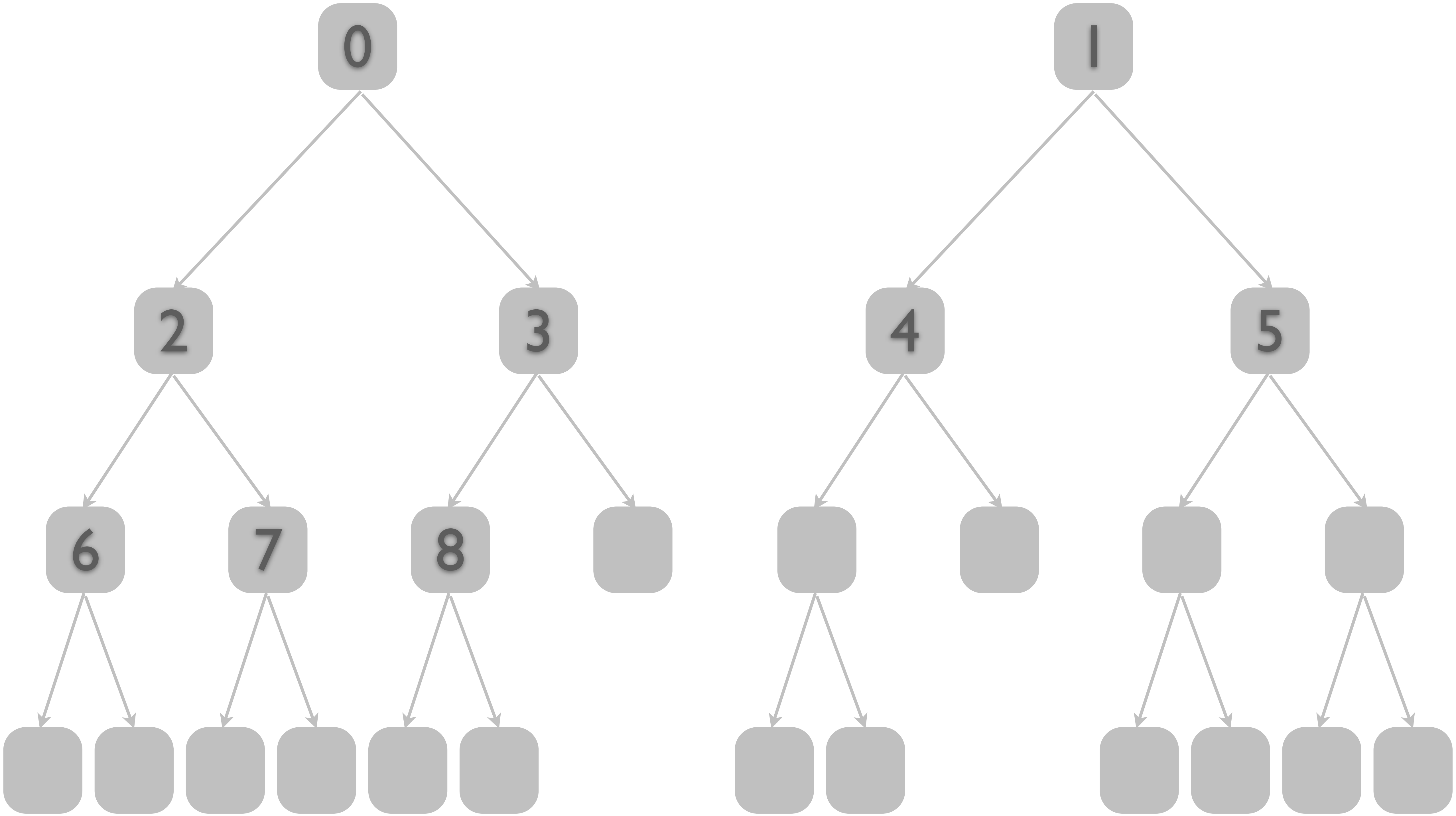


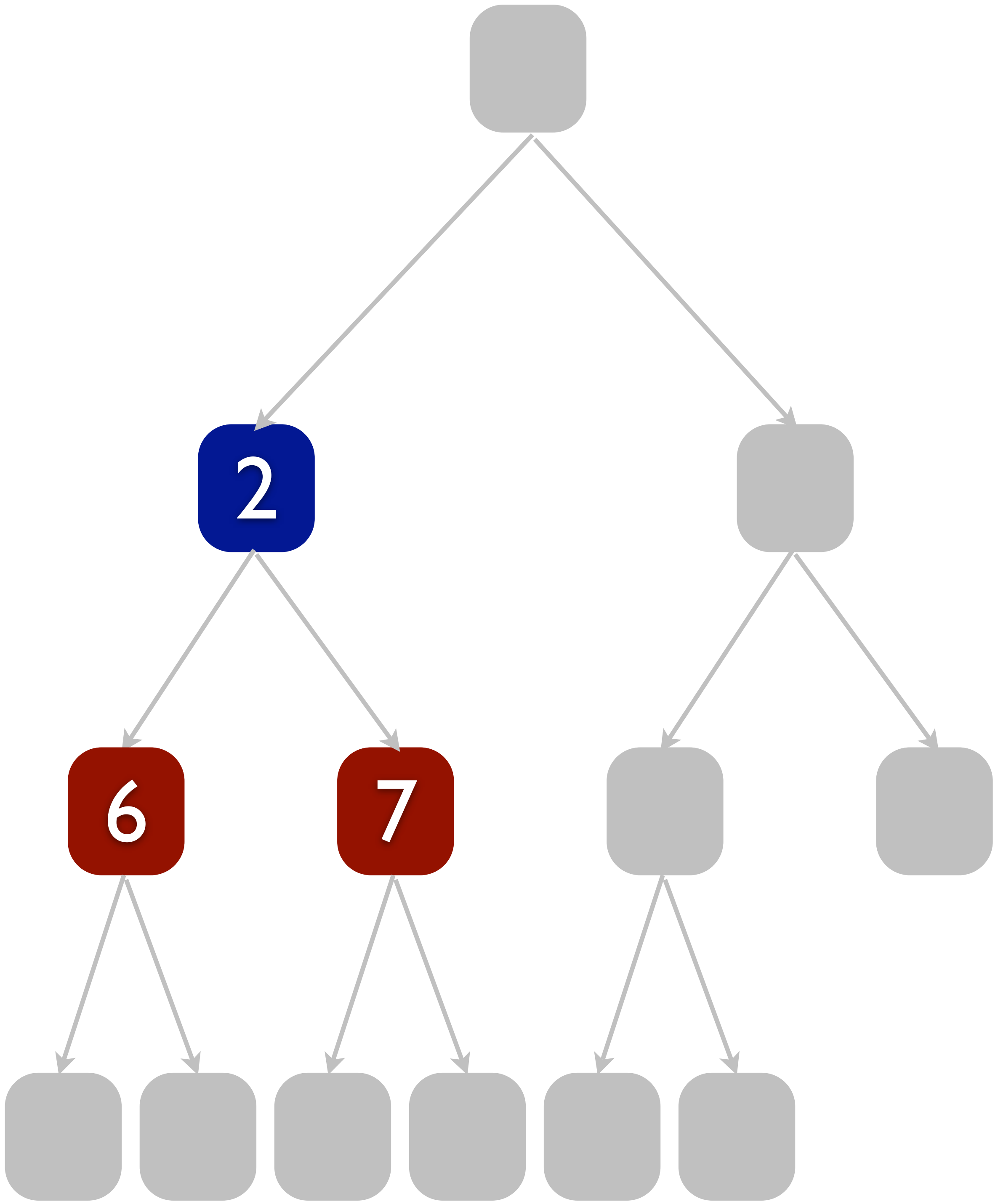
🟦 what to split

🟥 how to split

how to identify a vertex split?

Attempt 1

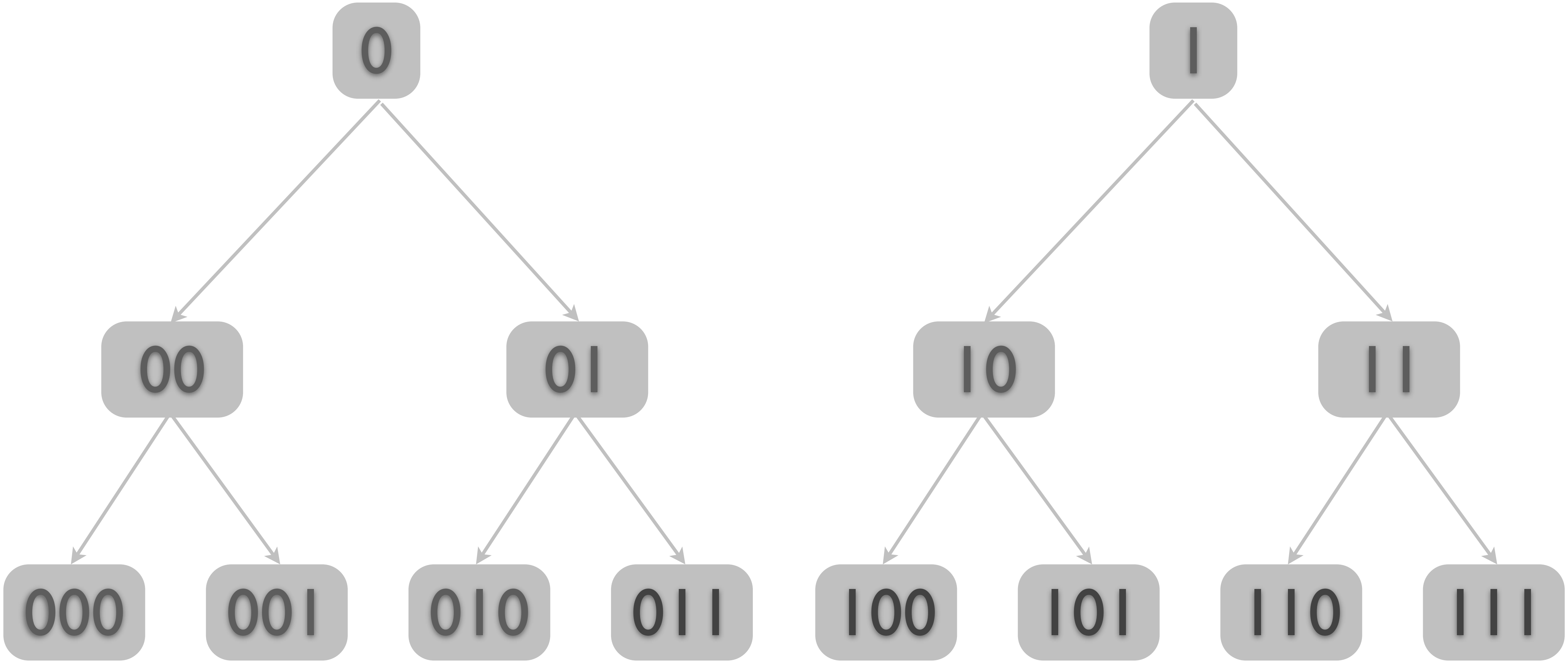




want to split vertex 2

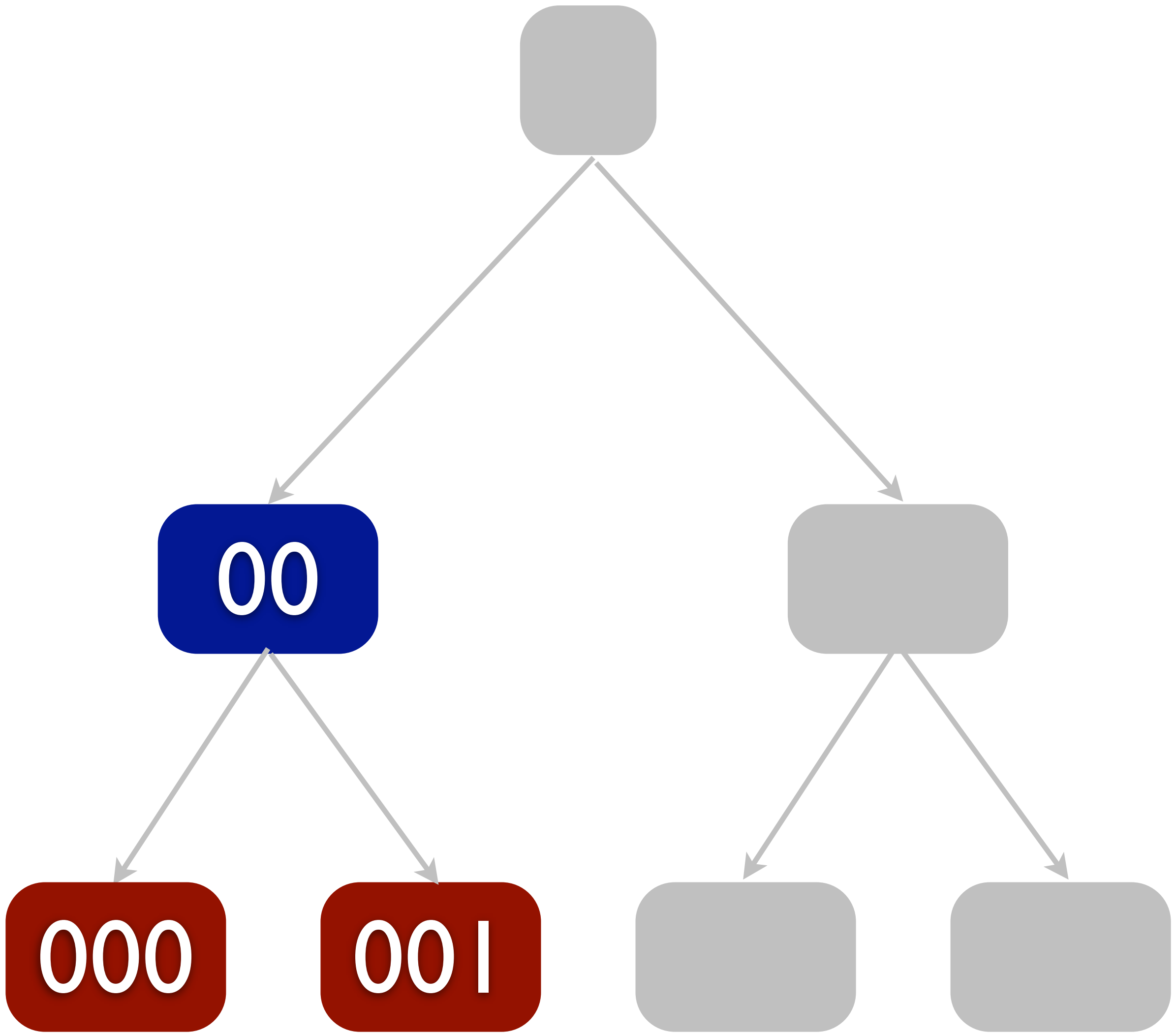
here is how to split, and
2 splits into 6 and 7

Attempt 2



Kim, Lee, "Truly selective refinement of progressive meshes,"

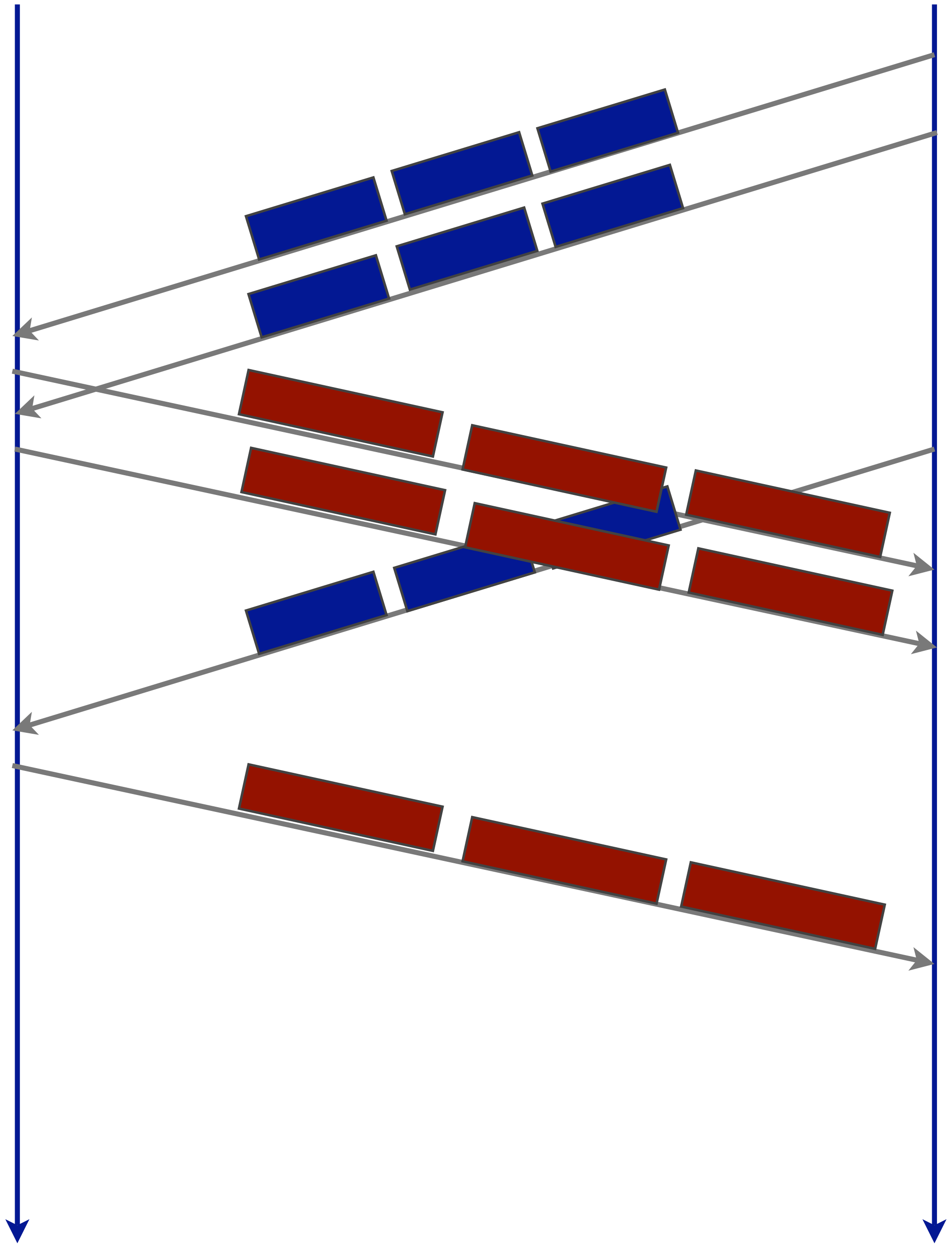
In Proceedings of Graphics Interface, pages 101–110, June 2001

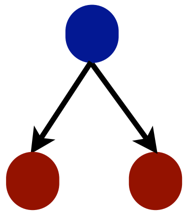


want to split vertex 00

here is how to split 00

Receiver-driven Approach

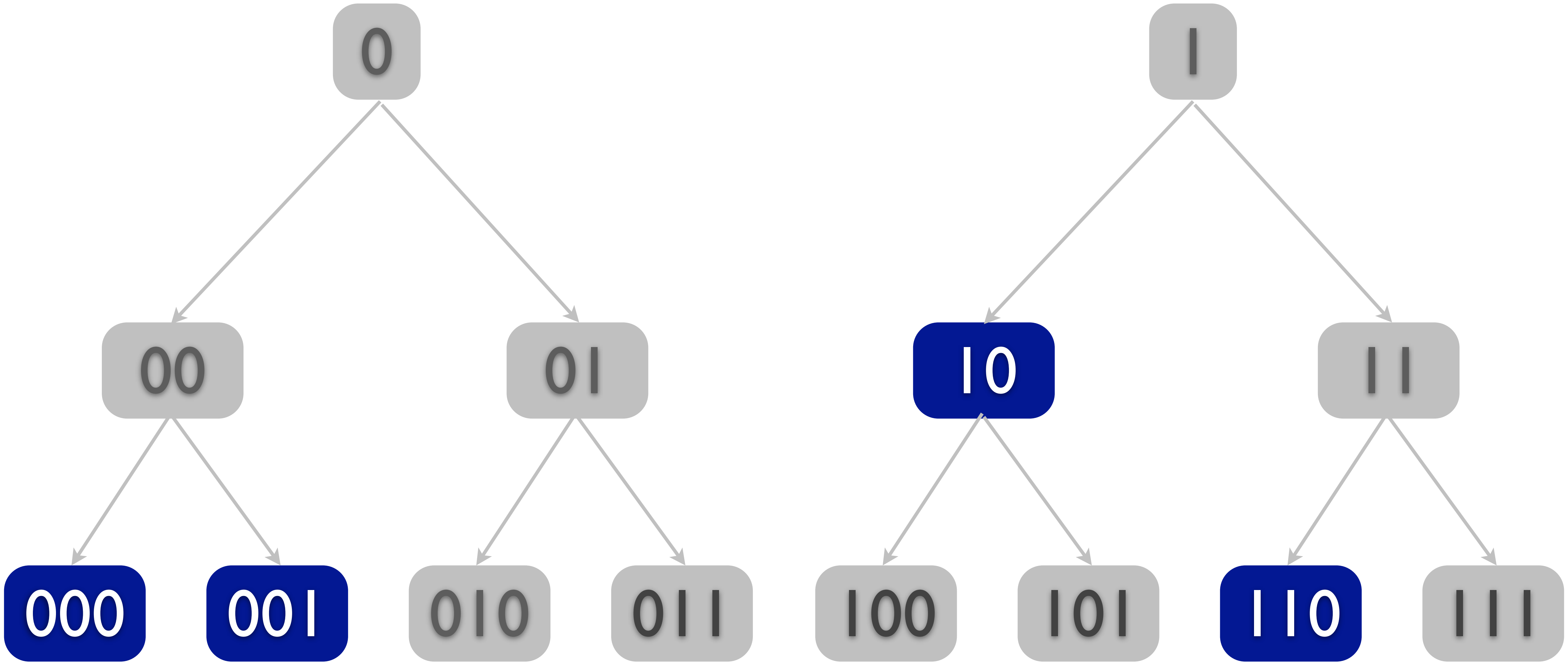




🟦 what to split

🟥 how to split

Encoding of vertex split IDs



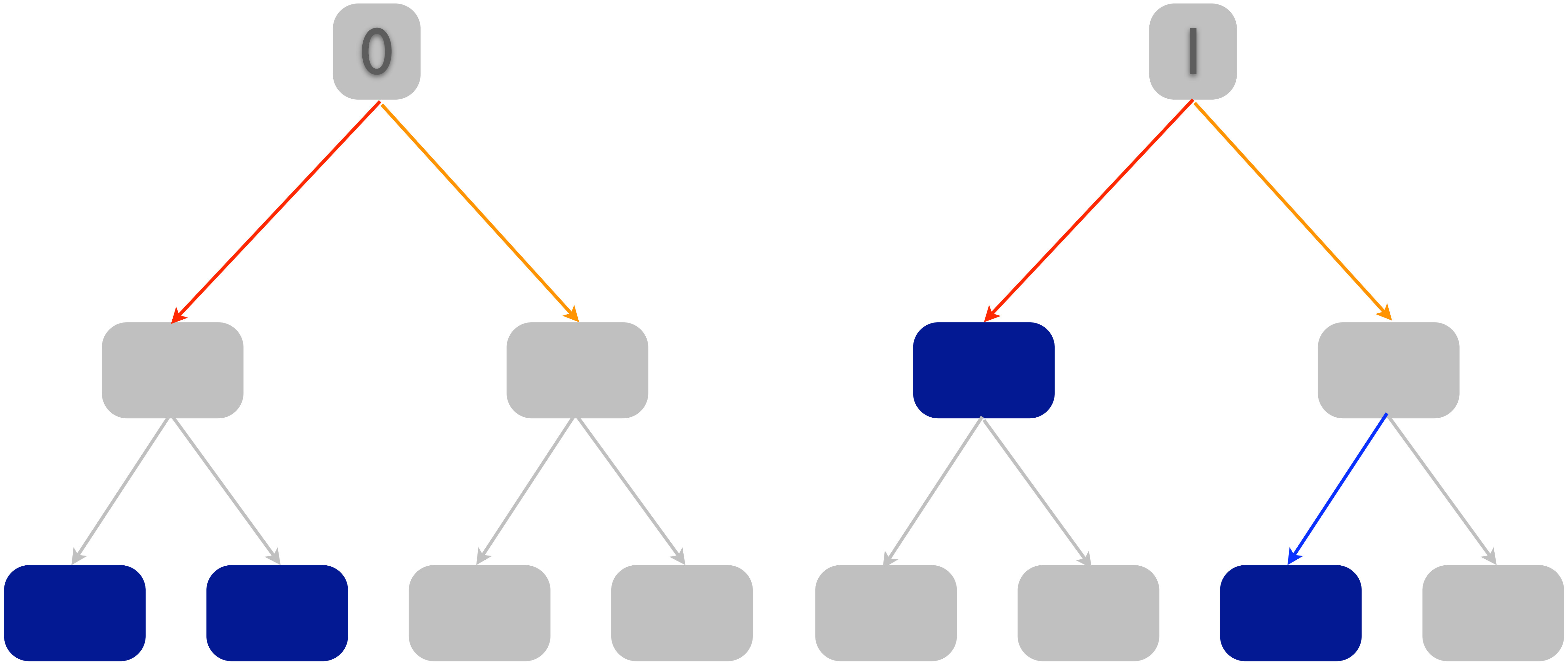
000 001 10 110

```
proc encode(T)

if no vertices to be split in T
    return 0

else
    return 1 + encode(T.left) + encode(T.right)
```

Encoding of vertex split IDs

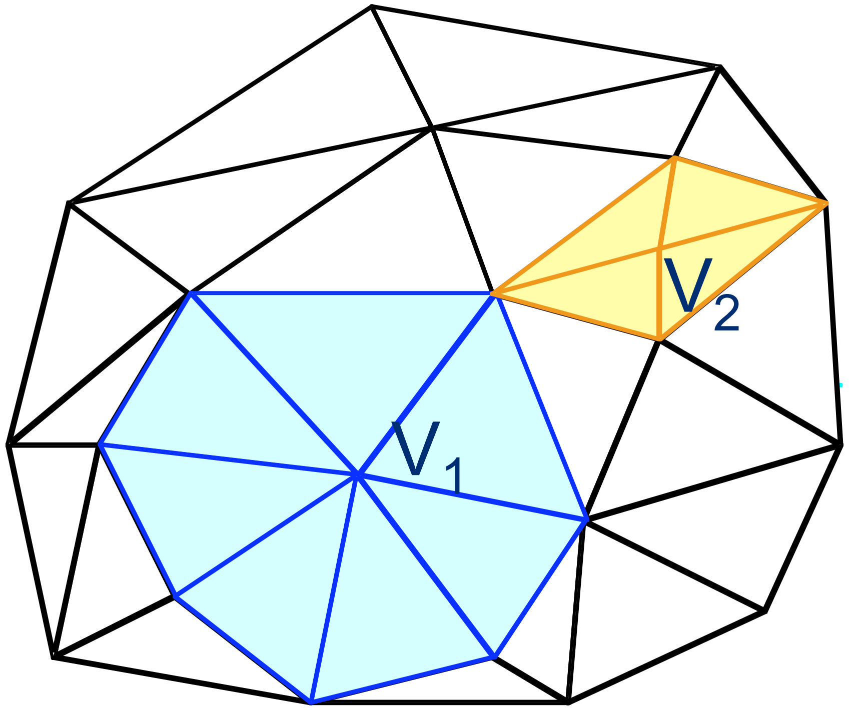


0 1 1001000 1 1001 1000

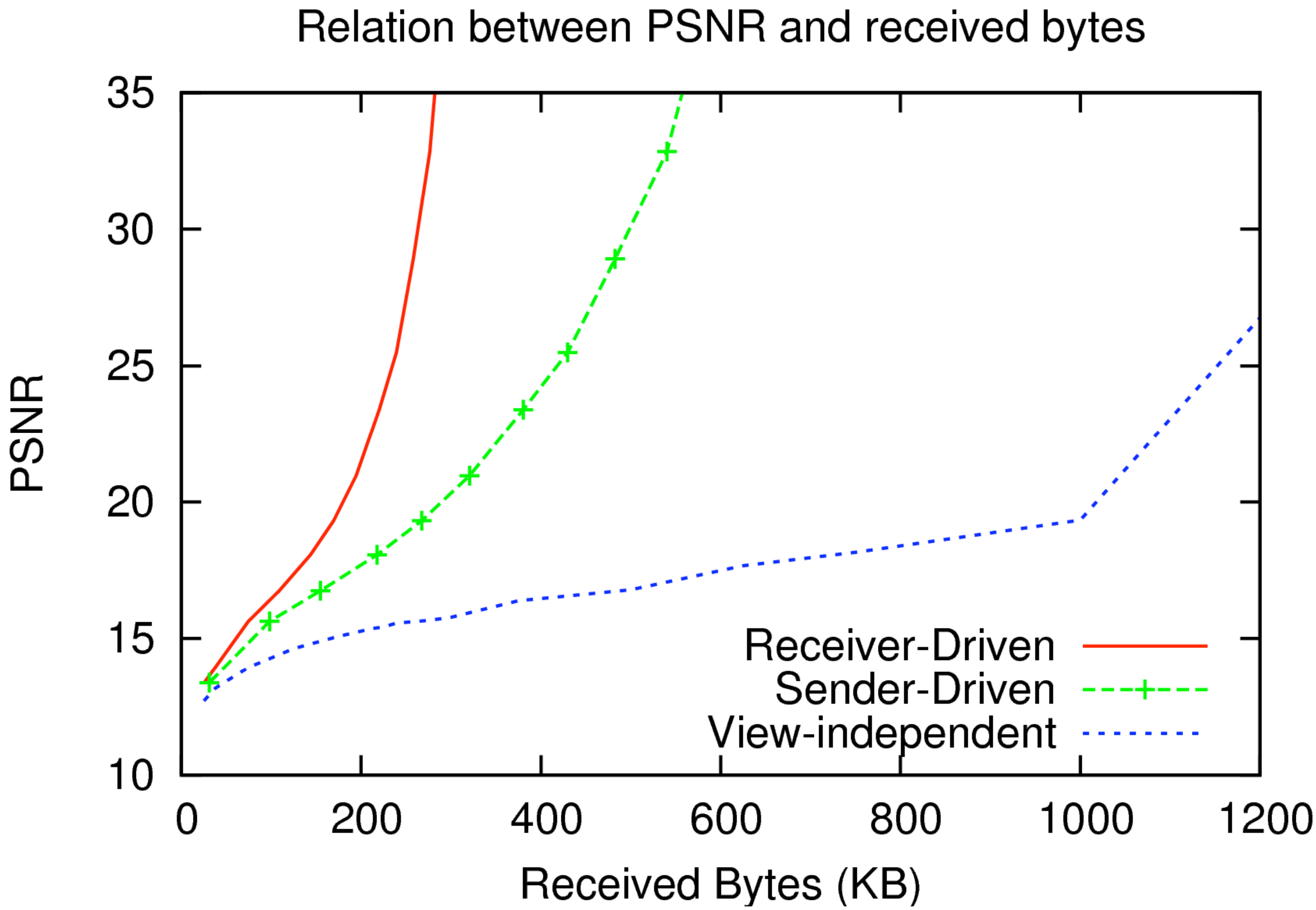
how to compute visibility + visual contributions?

(without possessing the complete mesh?)

Estimate with screen space area of vertices



	Sender-driven	**Receiver-driven**
send base mesh	1.4	1.13
decode IDs	-	1.55
search vertex split	1.85	1.85
determine visibility	0.41	-
update state	1.41	-
encode IDs	0.94	-
others	0.16	0.16
total	**6.17**	**4.69**


Relation between PSNR and received bytes
PSNR
35
30
25
20
15
10
0
200
400
600
800
1000
1200
Received Bytes (KB)
Receiver-Driven
Sender-Driven
View-independent

receiver-driven protocol alleviates the **computational** bottleneck at the sender.

the other bottleneck is **bandwidth**.

goal: reduce server overhead by retrieving vertex splits from other clients if possible

difficulty: need to quickly and efficiently determine who to retrieve the vertex splits from low server overhead

low response time

low message overhead

common P2P techniques:

1. build an overlay and push
2. use DHT to search for chunks
3. pull based on chunk availability

common P2P techniques:

~~1. build an overlay and push~~

~~2. use DHT to search for chunks~~

3. pull based on chunk availability

peer-to-peer file transfer:

a needed chunk is likely to be available in any peer

peer-to-peer video streaming:

a needed chunk is likely available from a peer that has watched the same segment earlier

(temporal locality)

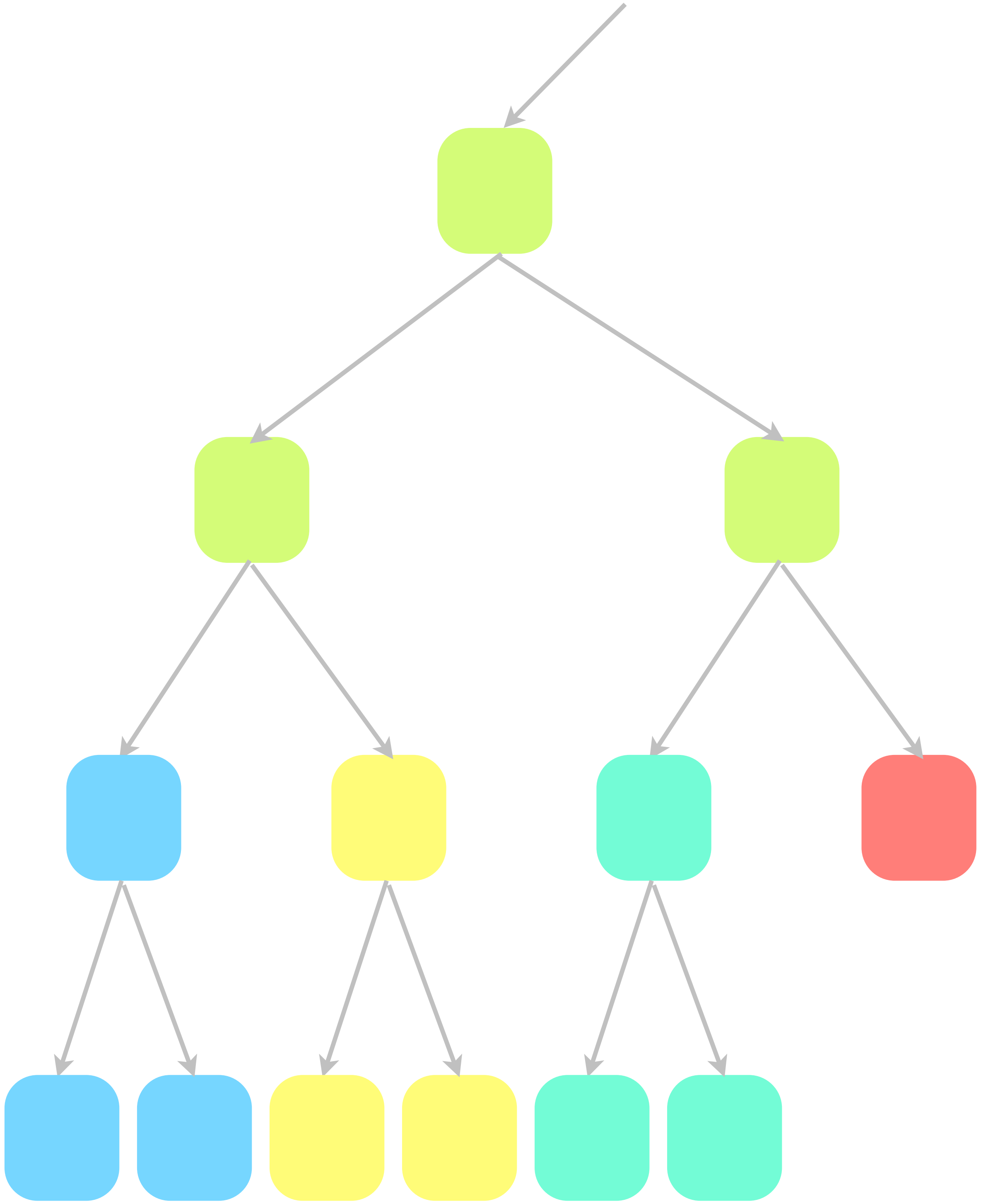
peer-to-peer mesh streaming

a needed chunk is likely available from a peer that is viewing the same region

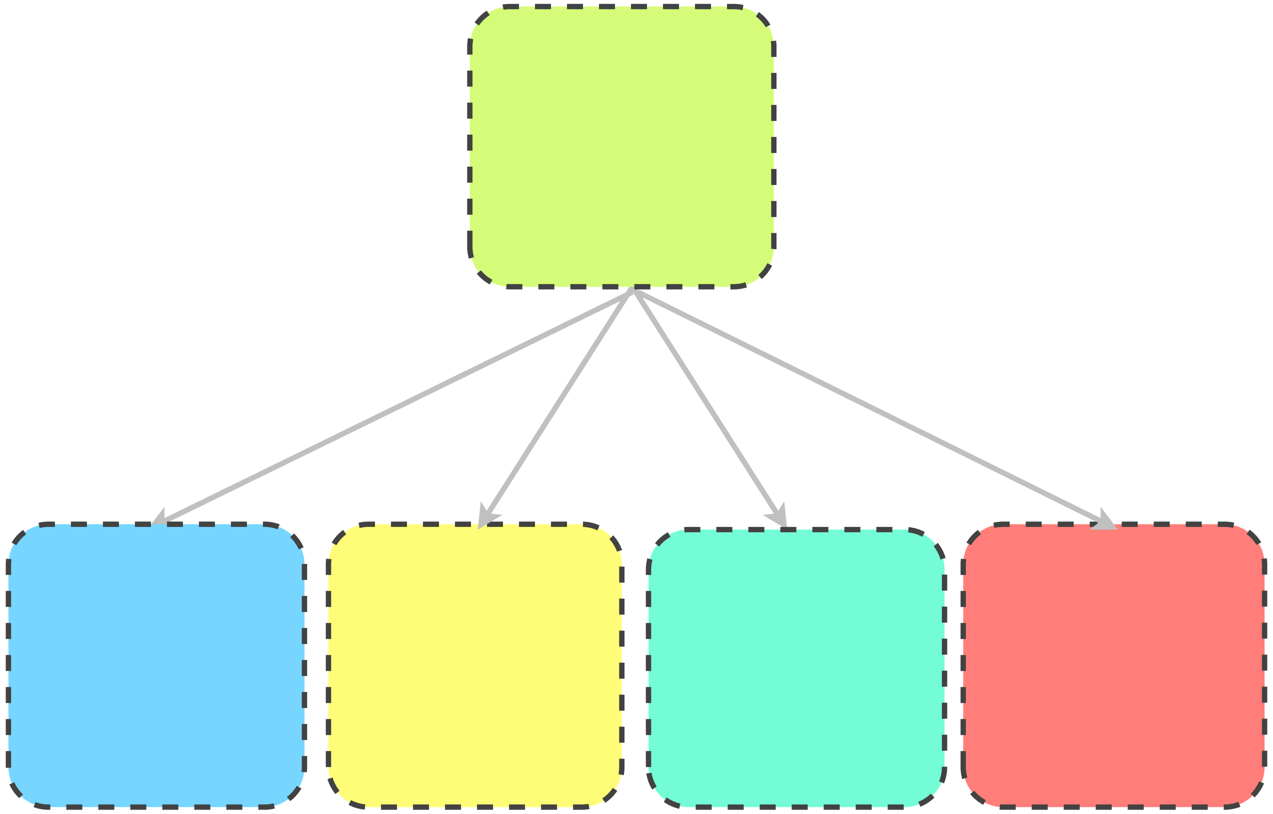
(spatial locality)

idea: exploit spatial locality to reduce message overhead.

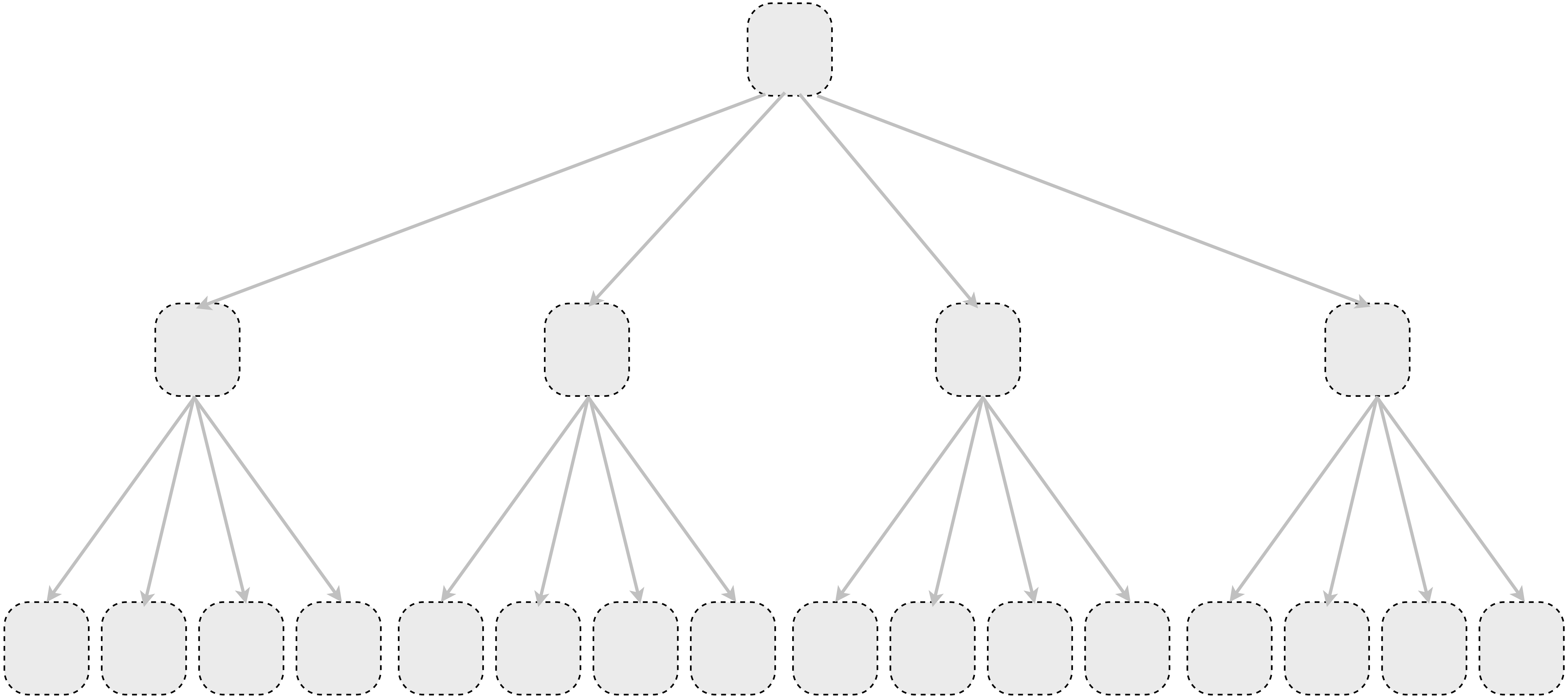
chunks



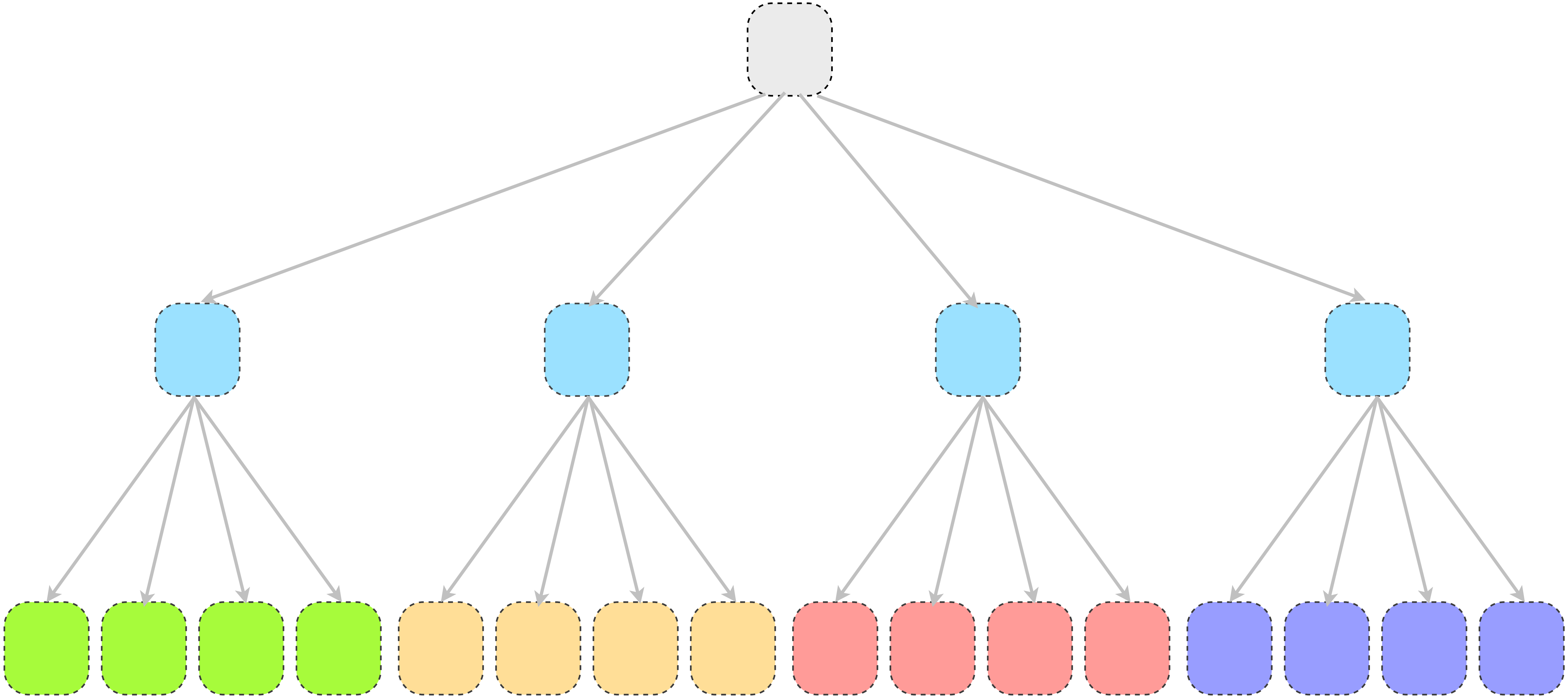
chunks



(1 chunk = 240 vertex splits)



groups

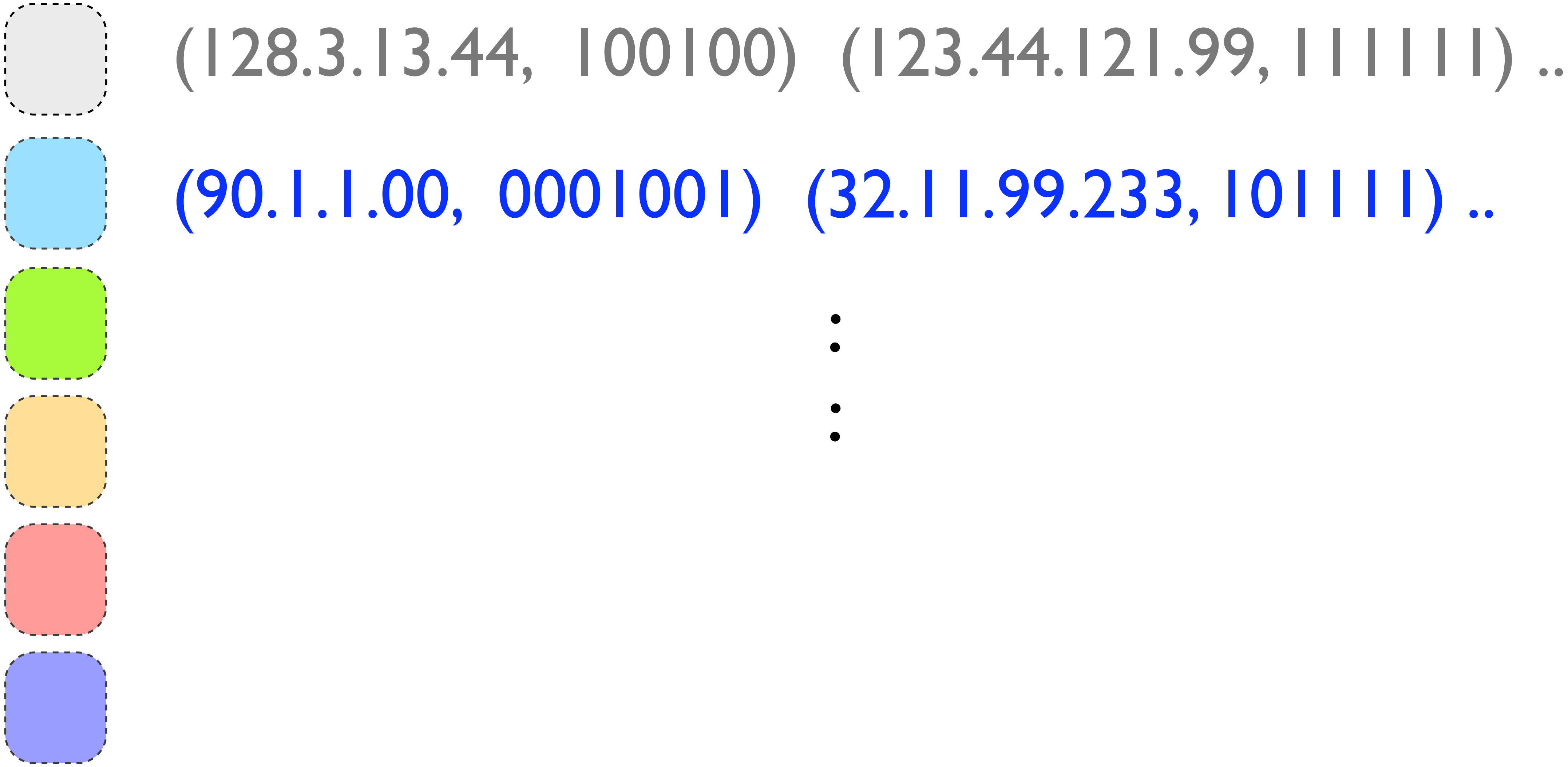


(1 group = 16 chunks)

Only exchange messages between peers that need chunks from the same group.

how the protocol works

server maintains a list of group members
for each group, and who possesses which chunk.



(128.3.13.44, 100100) (123.44.121.99, 111111) ..

(90.1.1.00, 0001001) (32.11.99.233, 101111) ..

:

:

client: “I want to view mesh M”

server sends :

(i)   base mesh

(ii) group members of the highest group.

(iii) what each member possess

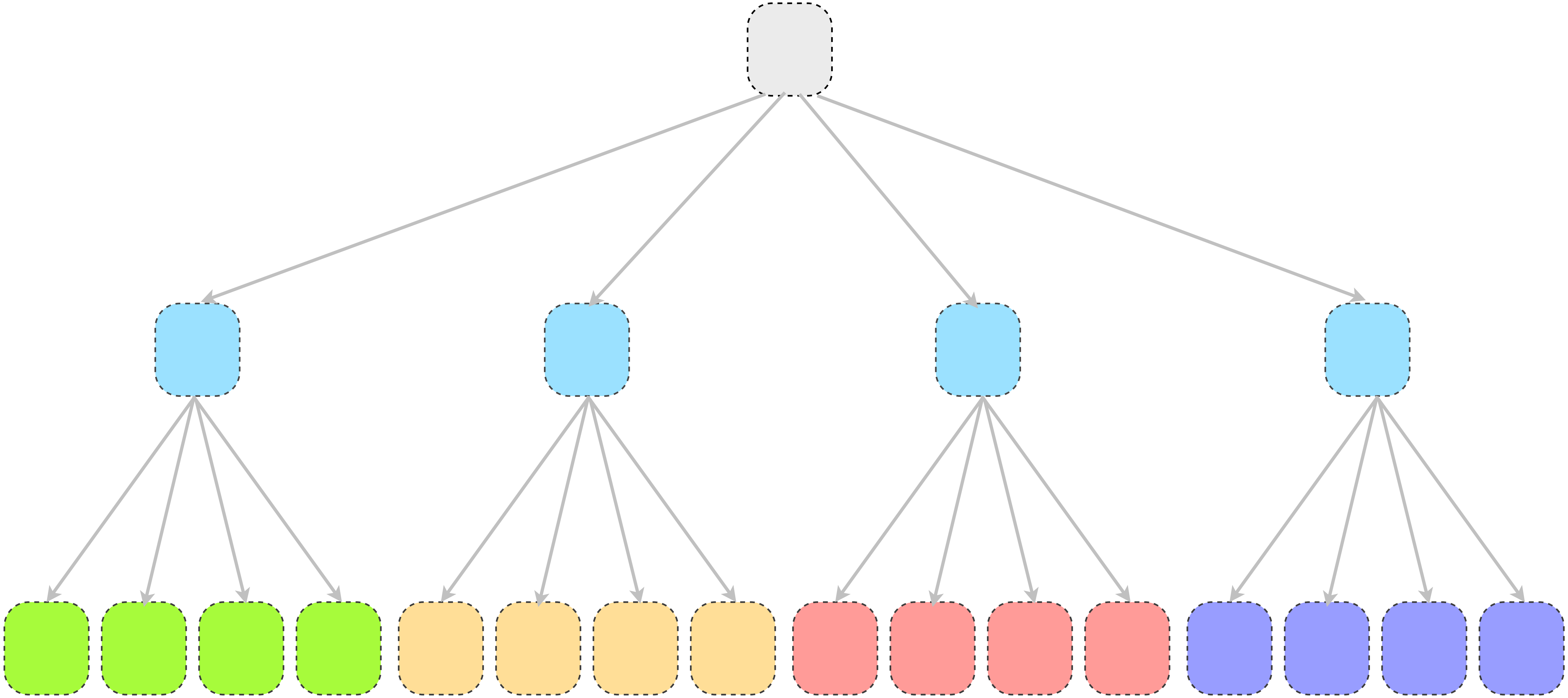
client decides which vertex splits (chunk) to refine

if some peer has that chunk, request from peer
else request chunk from server

peers inform server when they received a chunk

if a chunk in the next group can be decoded, server sends group members of the next group

groups



if too many group members, server sends only most recent subsets + some seeds

on-going work:

1. evaluation using user traces and simulator
2. other design parameters
3. further reduce the role of server

summary

receiver-driven design to reduce CPU cost

peer-to-peer design to reduce bandwidth cost

謝謝