Multimodal Synchronization in VR

Gerard Kim, Korea University

VR is multimodal

- VR as a media to deliver rich experience tends to be or will be multimodal relying and leveraging on many sensing and motor capabilities of the user
- VR interaction events
  - For input and output
  - One event - many modalities
    - The multimodal input or output for a single event must occur within “short” time duration (usually 100~300 msec) to be recognized as a single event by the computer or user → Synchronization issue

Issue 1

- Specifying the need for synchronization - content issue
  - Information model for MAR contents
    - Added to support for representing multimodal event/behavior
  - Simultaneity
  - Strict timing
  - Input vs. output
  - SMIL

SMIL

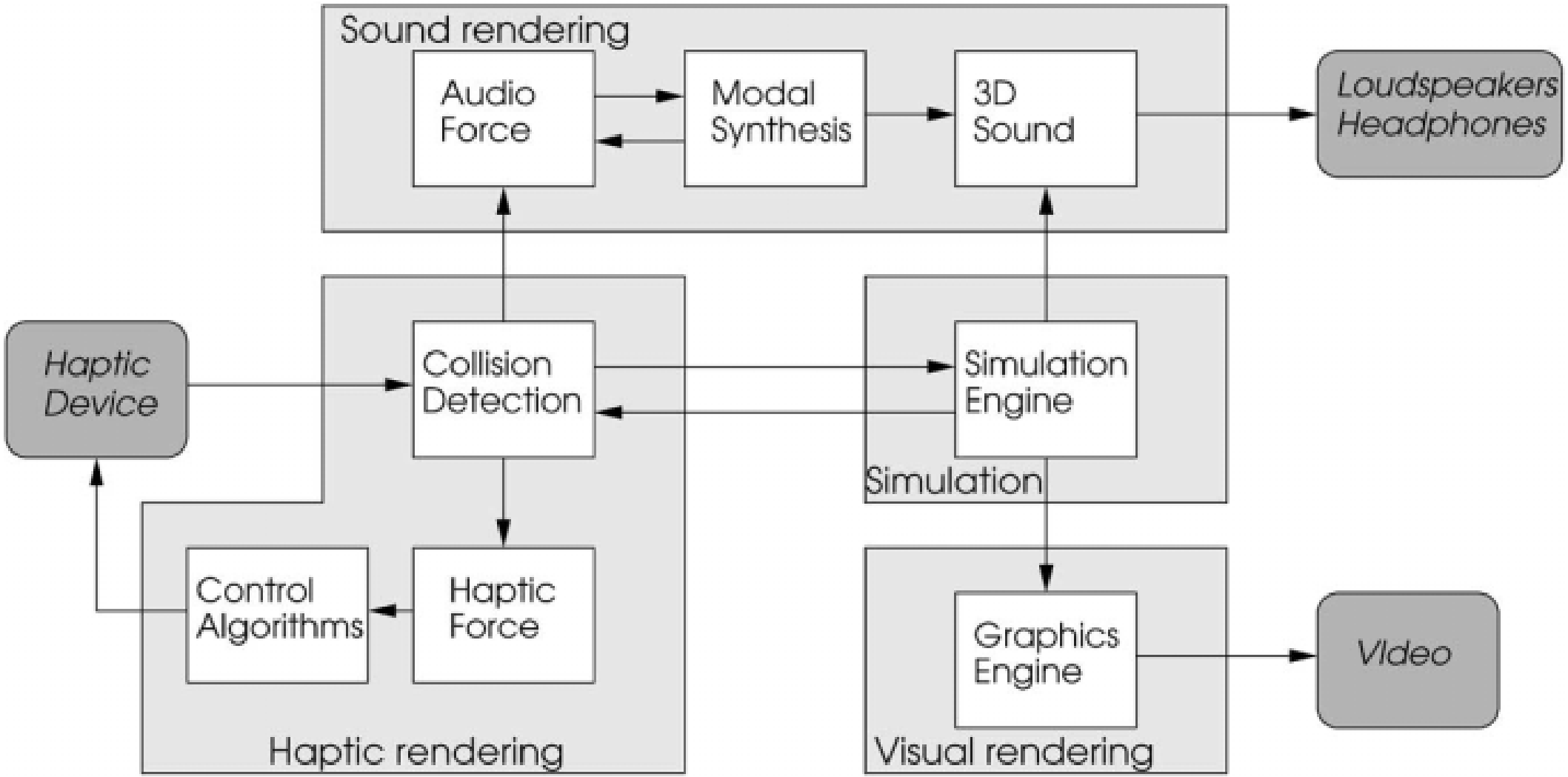
- SMIL (pronounced "smile") stands for "Synchronized Multimedia Integration Language" and defines scheduling ("Synchronized"), video, audio, images, text ("Multimedia"), multi-zone screen layout ("Integration") in an XML-based text file format ("Language").
- It is an open specification (royalty-free to use) created by the World-Wide Web Consortium.
- SMIL playlists may be scheduled to play at specific times and repeat intervals. This is the single greatest difference between SMIL and other XML-based multimedia markup languages.
  - Play at specific time each day
  - Play at a given day of week
  - Play at a specific time

```
<smil>
        <head />
        <body>
                <seq repeatCount="indefinite">
                        <video src="ad1_15s.mpg" />
                        <video src="ad2_30s.mpg" />
                </seq>
        </body>
</smil>
```

Issue 2

- How to achieve synchronization –browser/implementation issue
  - Performance benchmark (vs. theoretical conceptual simulation model)
    - Default ergnonmic requirement
  - Protocol/Methods in scripting/API for multimodal I/O

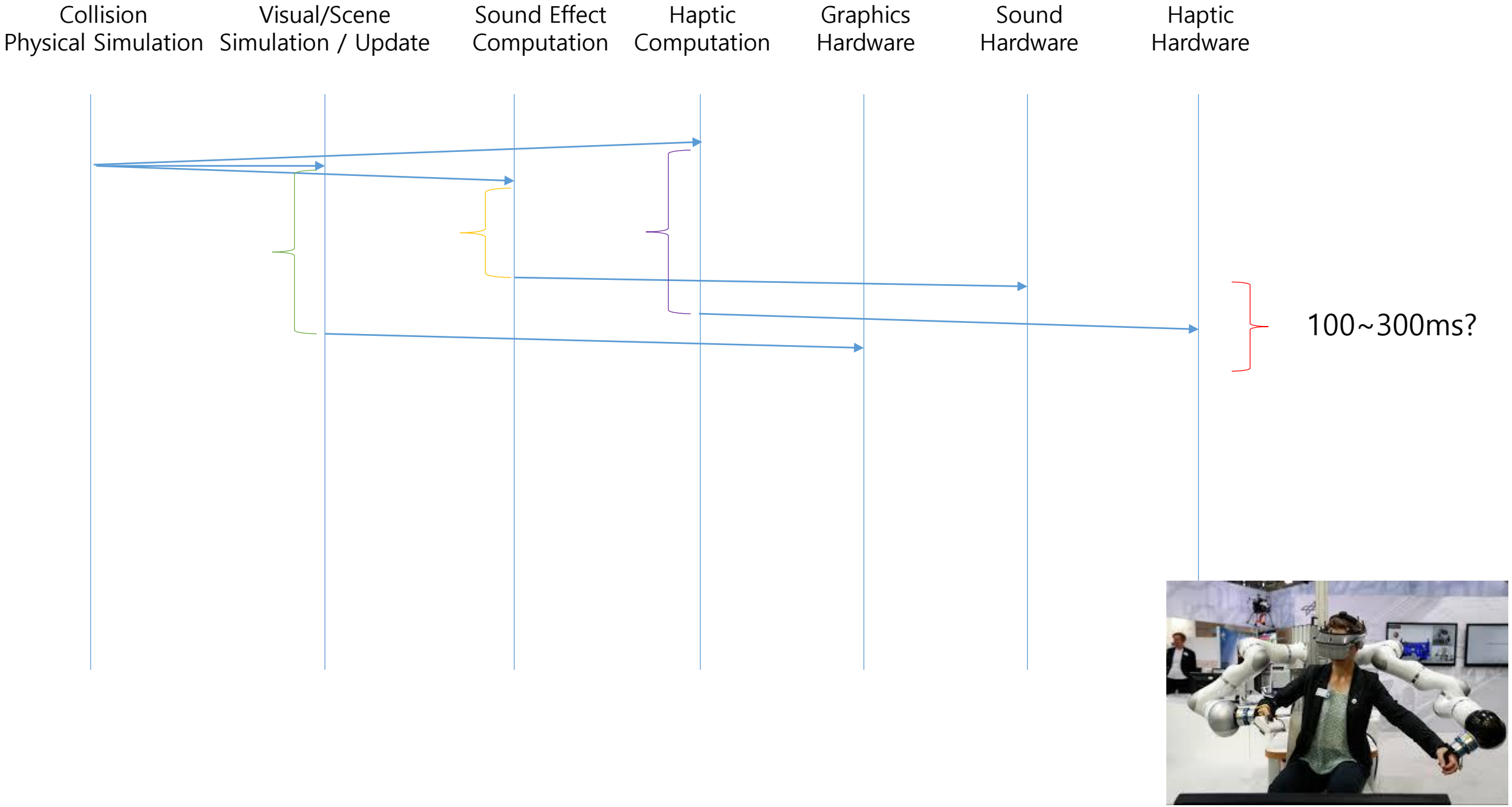
A Typical multimodal VR architecture (Avanzini and Crosato, 2006)



Usual approach – Best effort

- E.g.
  - Collision is detected by the physical simulator
    - Visual scene is simulated and updated as fast as possible
      - Then it is sent to visual renderer (graphics hardware) as fast as possible
    - Sound effect is created/computed/retrieved as fast as possible
      - Then it is sent to sound generator as fast as possible
    - Haptic feedback is computed as fast as possible
      - Then it is sent to the haptic device as fast as possible

- Q: Will all output be synchronized within the time threshold?


Collision
Physical Simulation
Visual/Scene
Simulation / Update
Sound Effect
Computation
Haptic
Computation
Graphics
Hardware
Sound
Hardware
Haptic
Hardware
100~300ms?

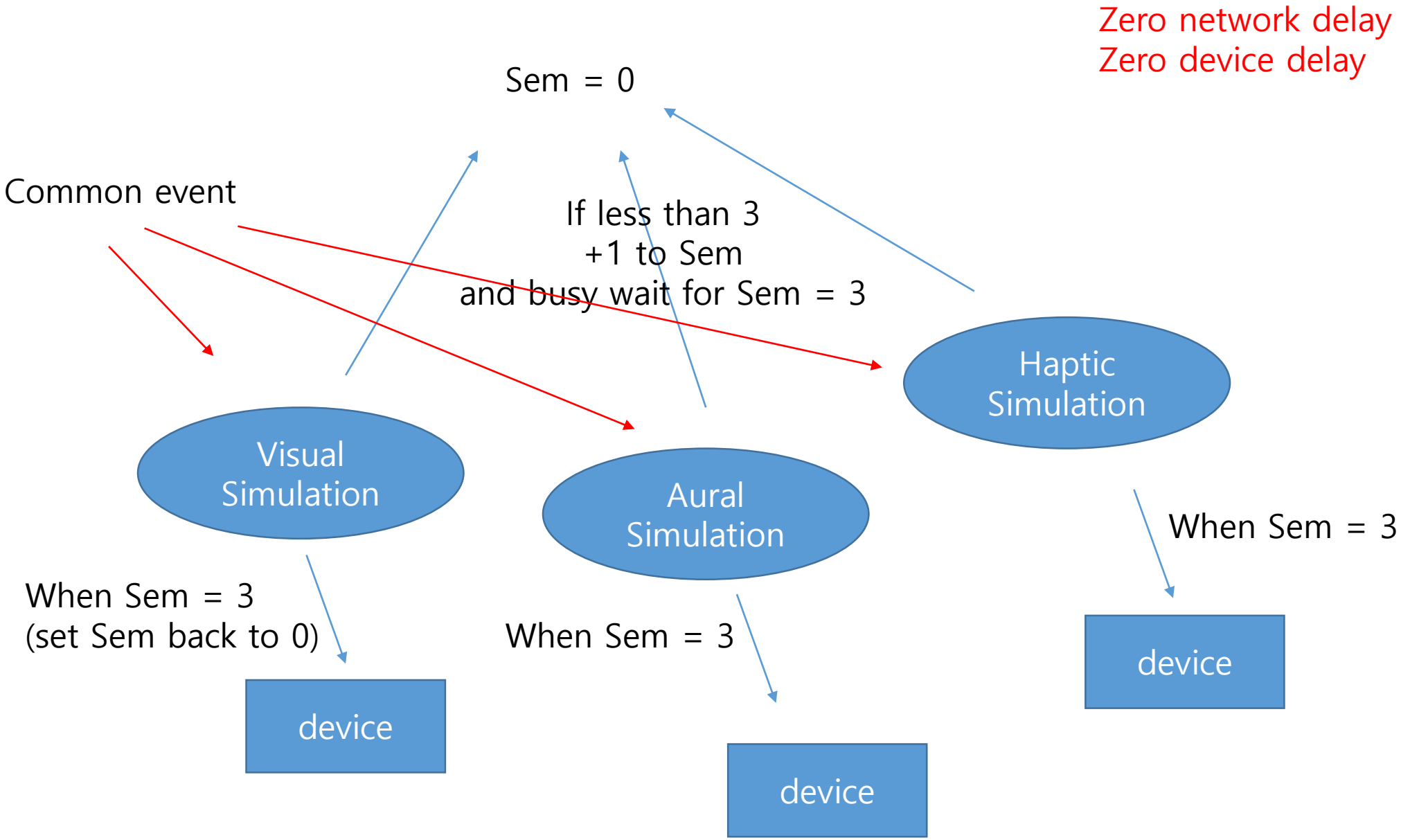
Usual approach – Best effort

- Best effort will work e.g. when
  - Model is not complex and the hardware/software is relatively very fast
  - We usually disregard hardware response time
  - Visual/Aural/Haptic rendering time often changes dynamically depending on the scene situation
  - We can expect that future VR will require much more processing load on each of these simulation and rendering threads
    - Best effort will someday not work properly

movie

Semaphores, Messages, Genlock, Software lock …

- Let's have these variable processes/threads exchange messages (or apply existing synchronization schemes) to align output (or input) timing
  - Define sem = 0
  - Each thread when finished with simulation and computation and ready to output to hardware will increment the sem by 1
    - Assume hardware will response immediately (for now)
  - If sem is equal to total number of threads (e.g. 3) then output to respective device
  - Otherwise thread waits and spins checking the value of sem
  - When outputs are made, the sem is set back to 0


Zero network delay
Zero device delay
Sem = 0
Common event
If less than 3
+1 to Sem
and busy wait for Sem = 3
Haptic Simulation
Visual Simulation
Aural Simulation
When Sem = 3
When Sem = 3
(set Sem back to 0)
When Sem = 3
device
device
device