Architecture and Mechanisms for High Quality Streaming Multimedia

Tze Sing Eugene Ng

Carnegie Mellon University

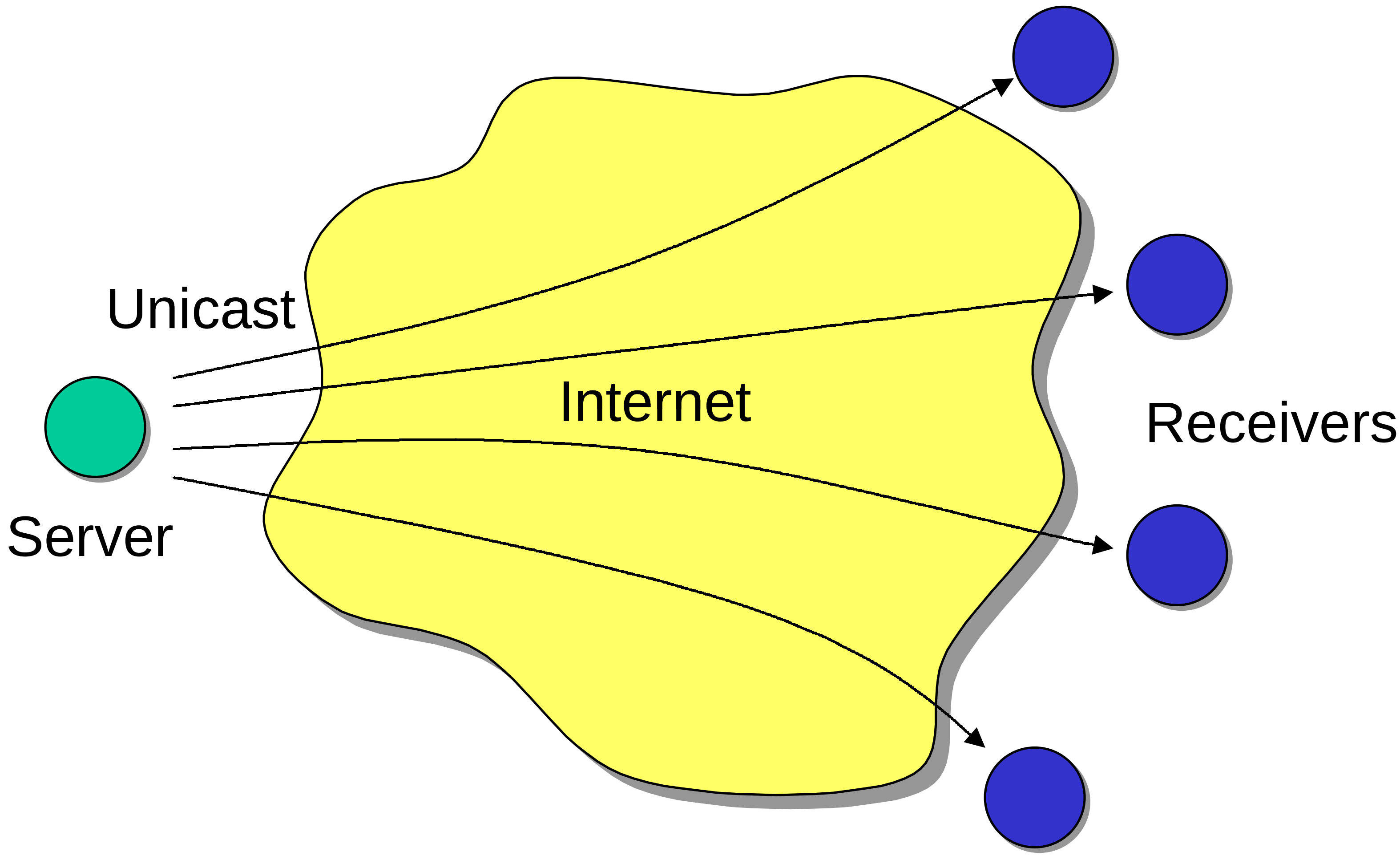
Joint work with Katie Guo, Markus Hofmann,

Sanjoy Paul, and Hui Zhang

Outline

- Motivation
- Assumptions
- Project goals
- Architecture overview
- Basic mechanisms
- Related work
- Conclusion

Current Streaming Architecture



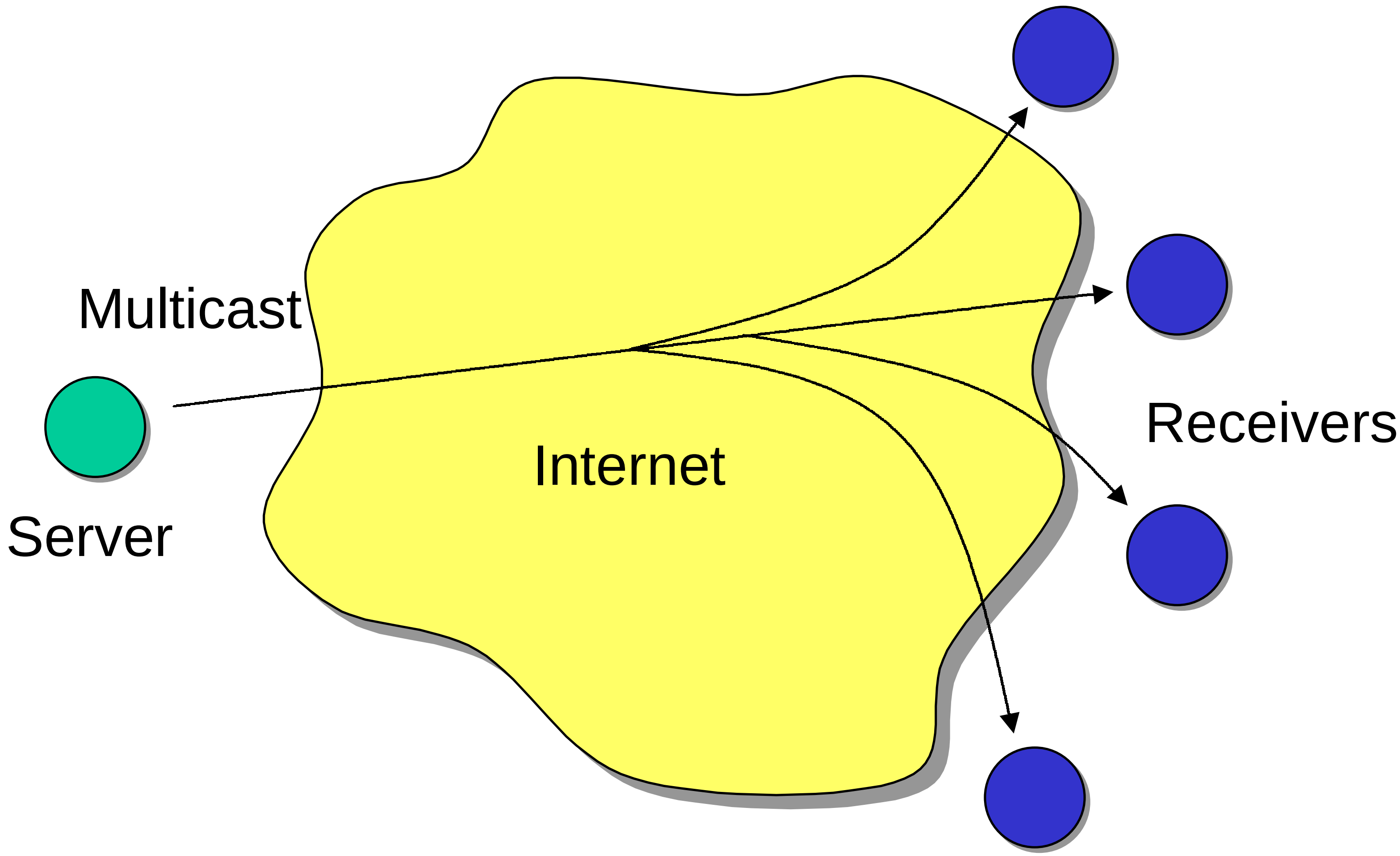
Problems with Current Architecture

- Content provider
  - Server load increases linearly with the number of receivers
- Receiver
  - High start-up latency
  - Unpredictable playback quality
  - Poor performance with VCR operations
- ISP
  - Streaming multimedia flows lead to serious network congestion problems

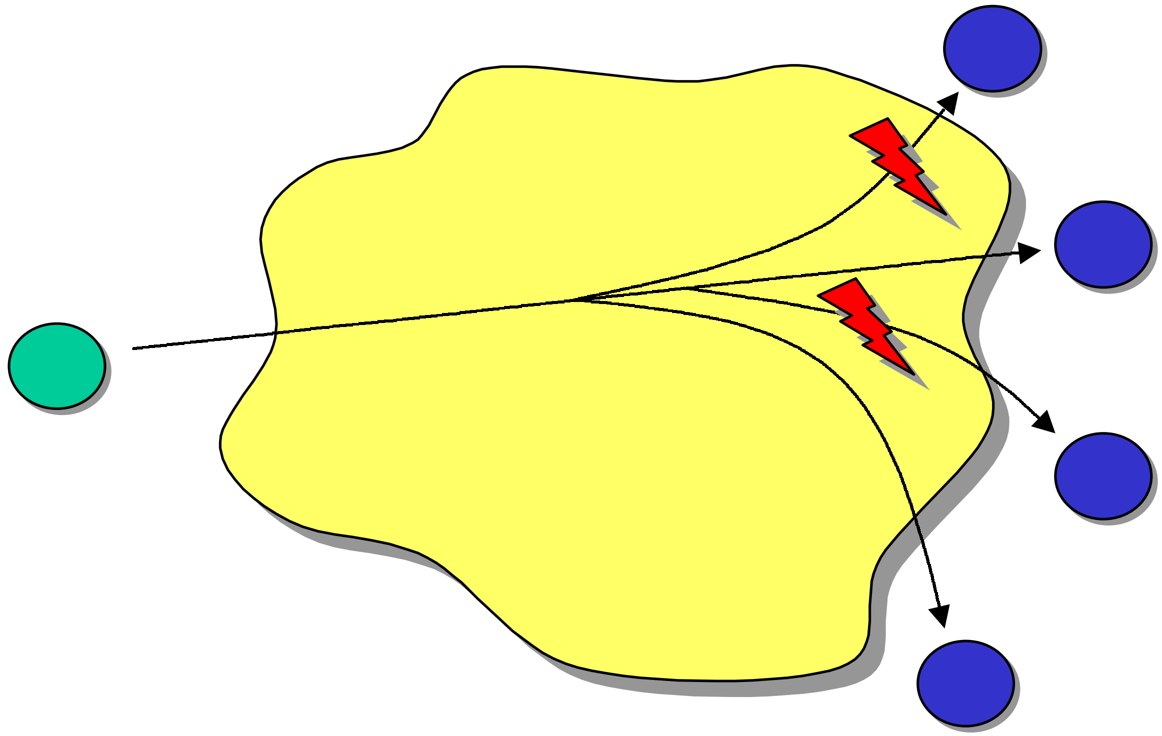
Example: News Coverage of President Clinton's Testimony

- CNN audio and video quality became unbearable for most people at around 1:00 pm on August 17
- Link to video stream removed from CNN by 1:15 pm
- Other news servers were also unreachable

IP Multicast

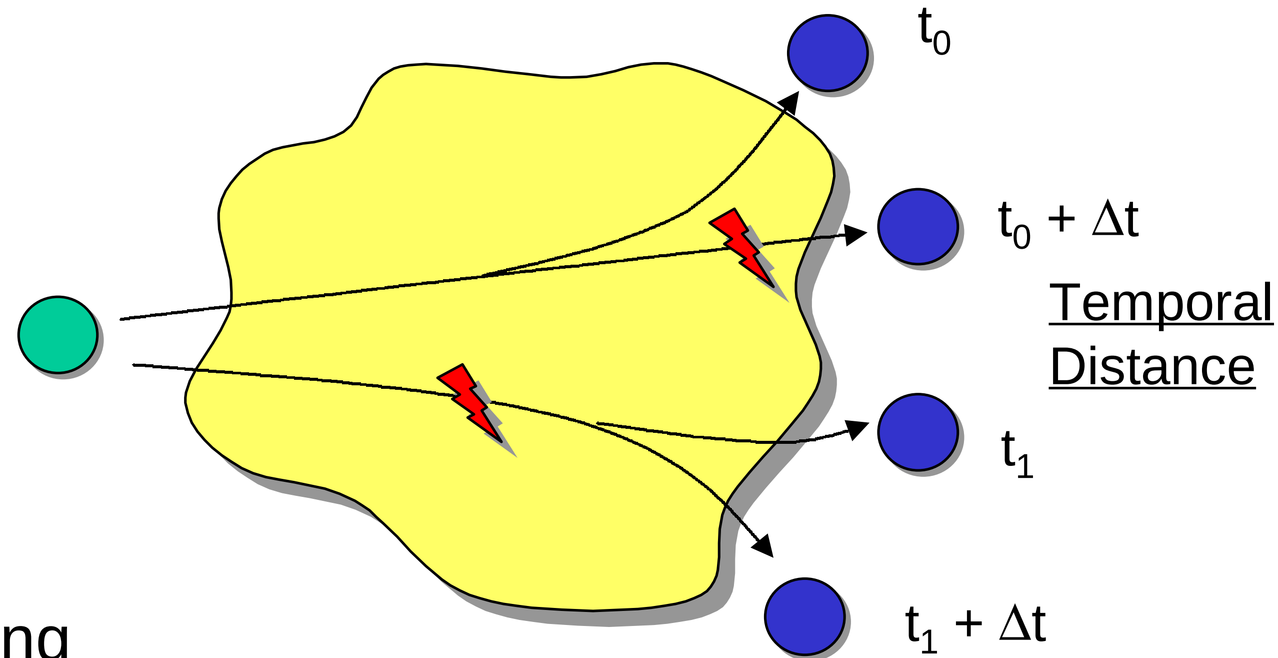


Multicast for Live Broadcast



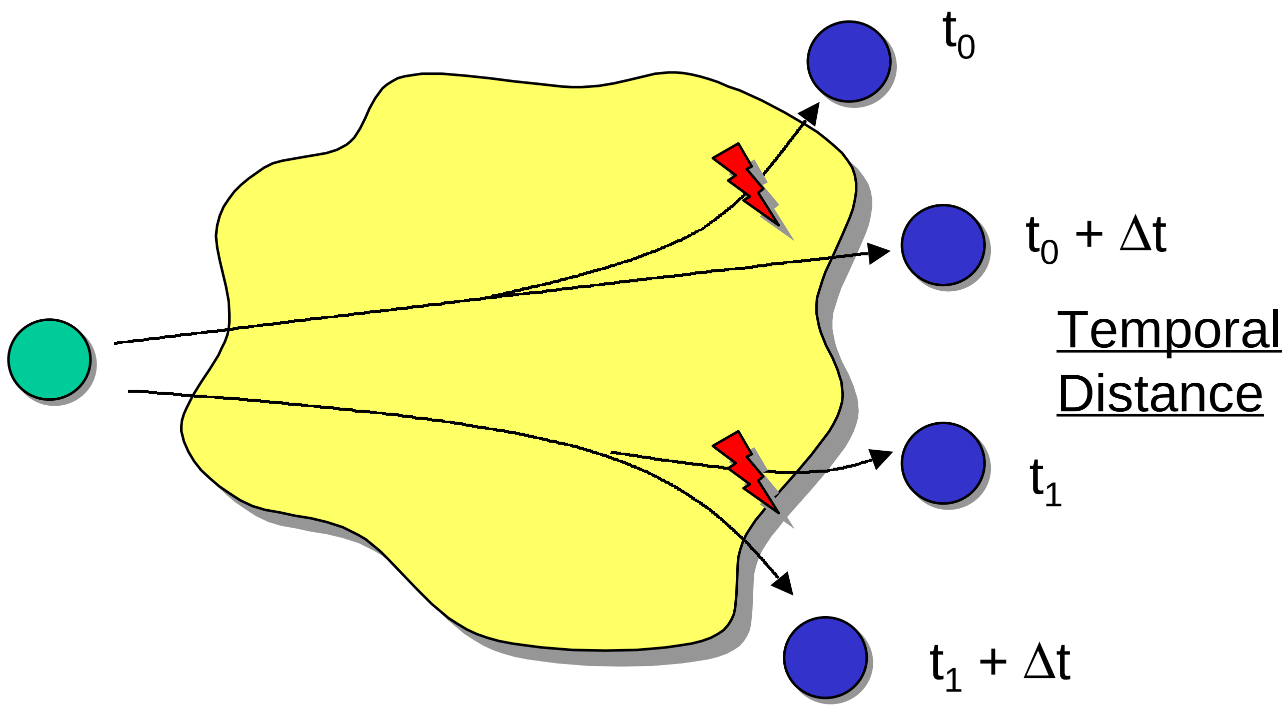
- Reduces both server load and network load
- Does not improve quality of service to receivers

Multicast for On-Demand Streaming



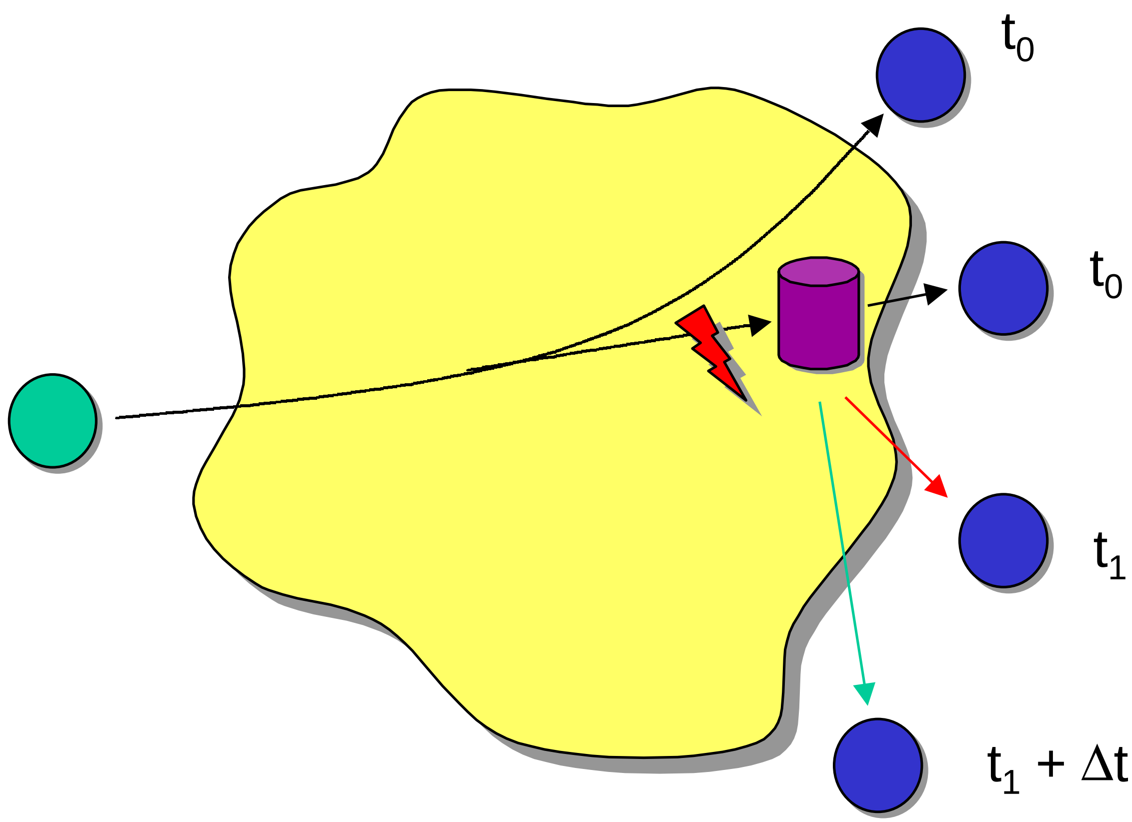
- Batching
  - Requests arriving within a time window Δt are batched together and are served by one multicast session
- Reduces both server load and network load
- Does not improve quality of service to receivers
  - Start-up latency increases

A Fundamental Problem with Multicast



- Multicast works best when receivers are homogeneous and synchronous
- In reality, receivers are heterogeneous and asynchronous

Solution: Buffer, Buffer, More Buffer



- Buffering can mask both the heterogeneity and asynchrony of receivers

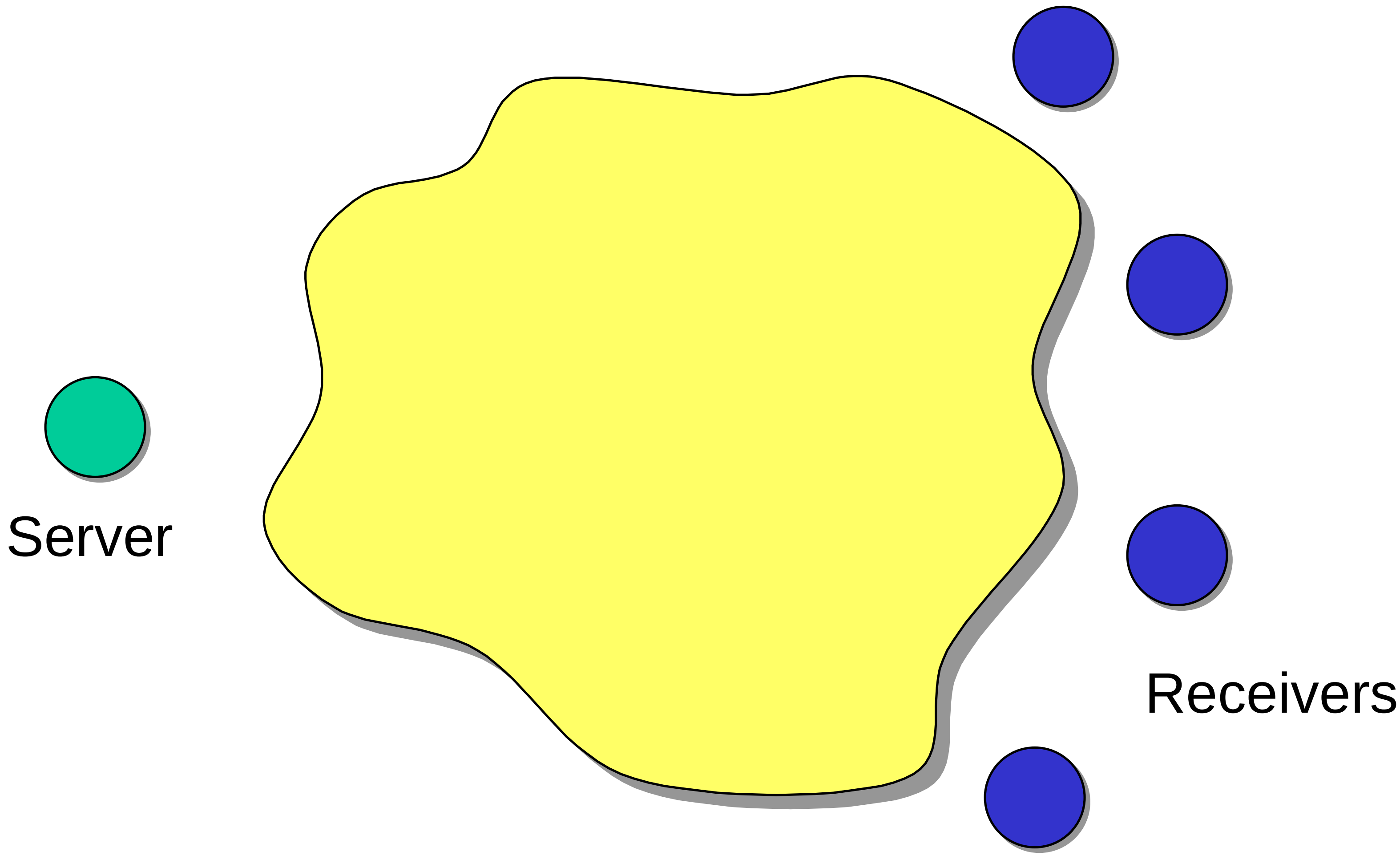
Assumptions

- Application environment
  - Live broadcasts
  - On-demand long streams
  - On-demand short clips
  - VCR operations allowed
- Network environment
  - Assume minimal QoS support
  - Can take advantage of underlying QoS if available
- Infeasible to replicate all streaming multimedia objects in their entirety

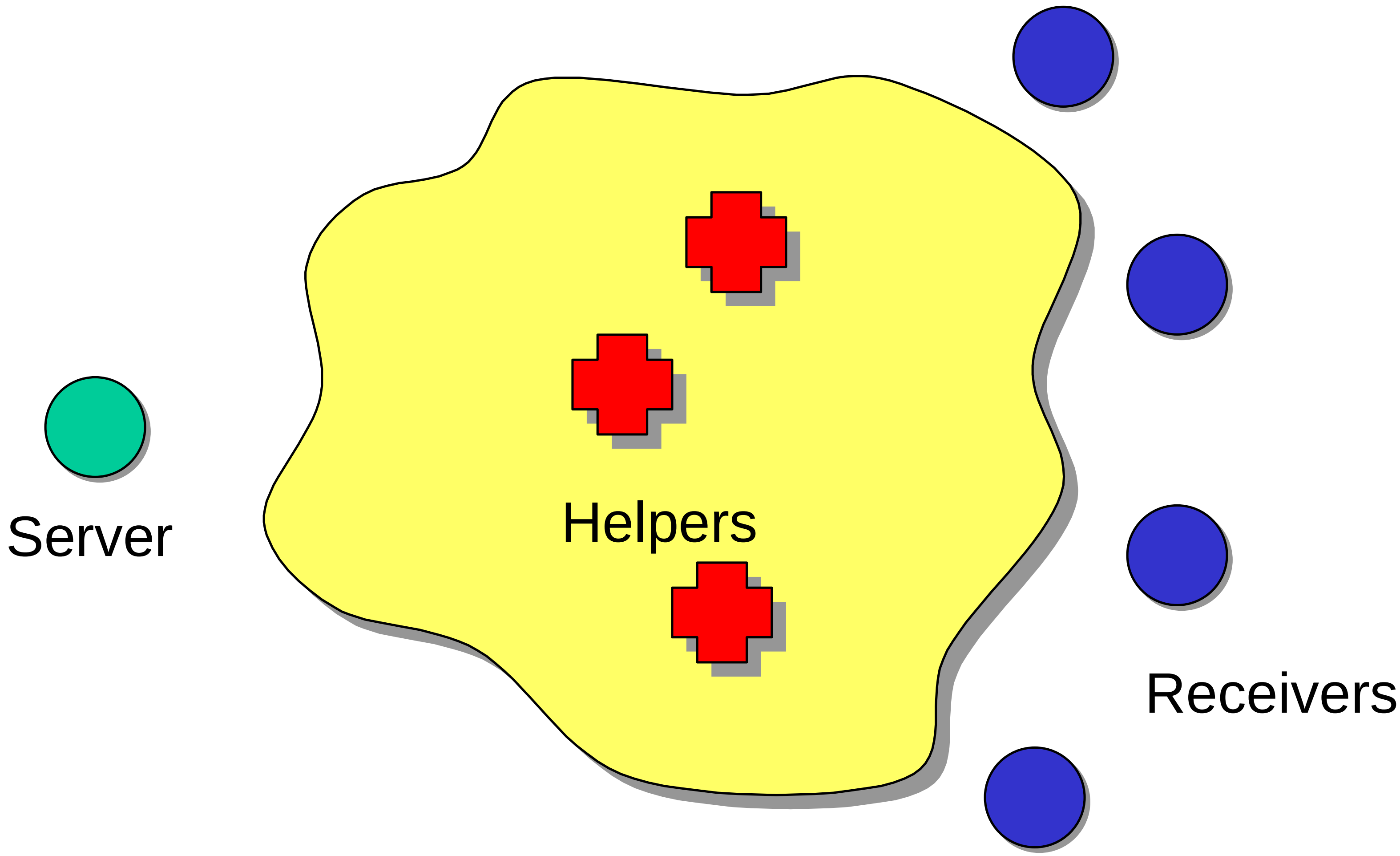
Project Goals

- Reduce start-up latency
- Improve playback quality
- Improve performance for VCR operations
- Reduce server load
- Reduce network load

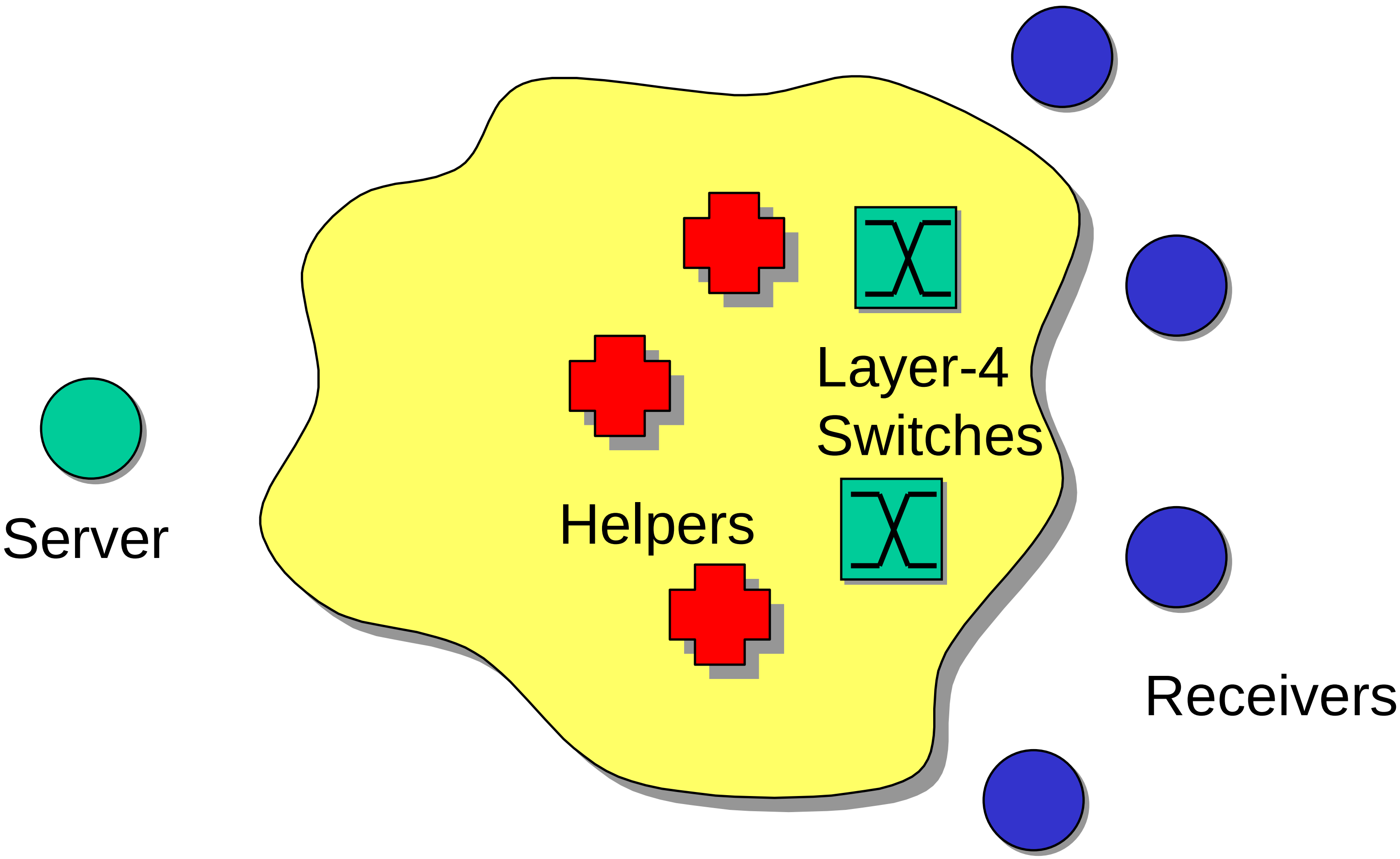
New Streaming Architecture



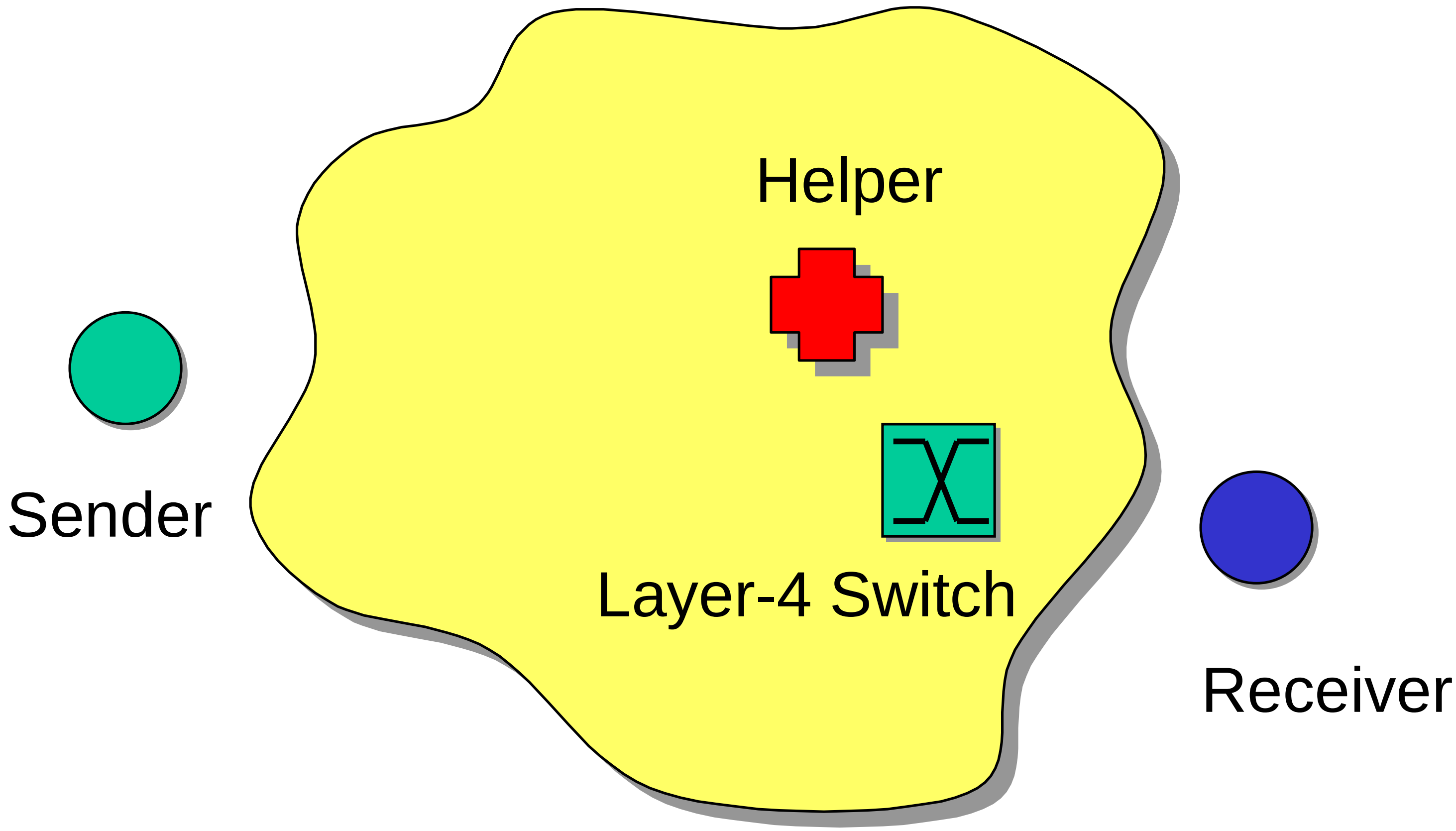
New Streaming Architecture



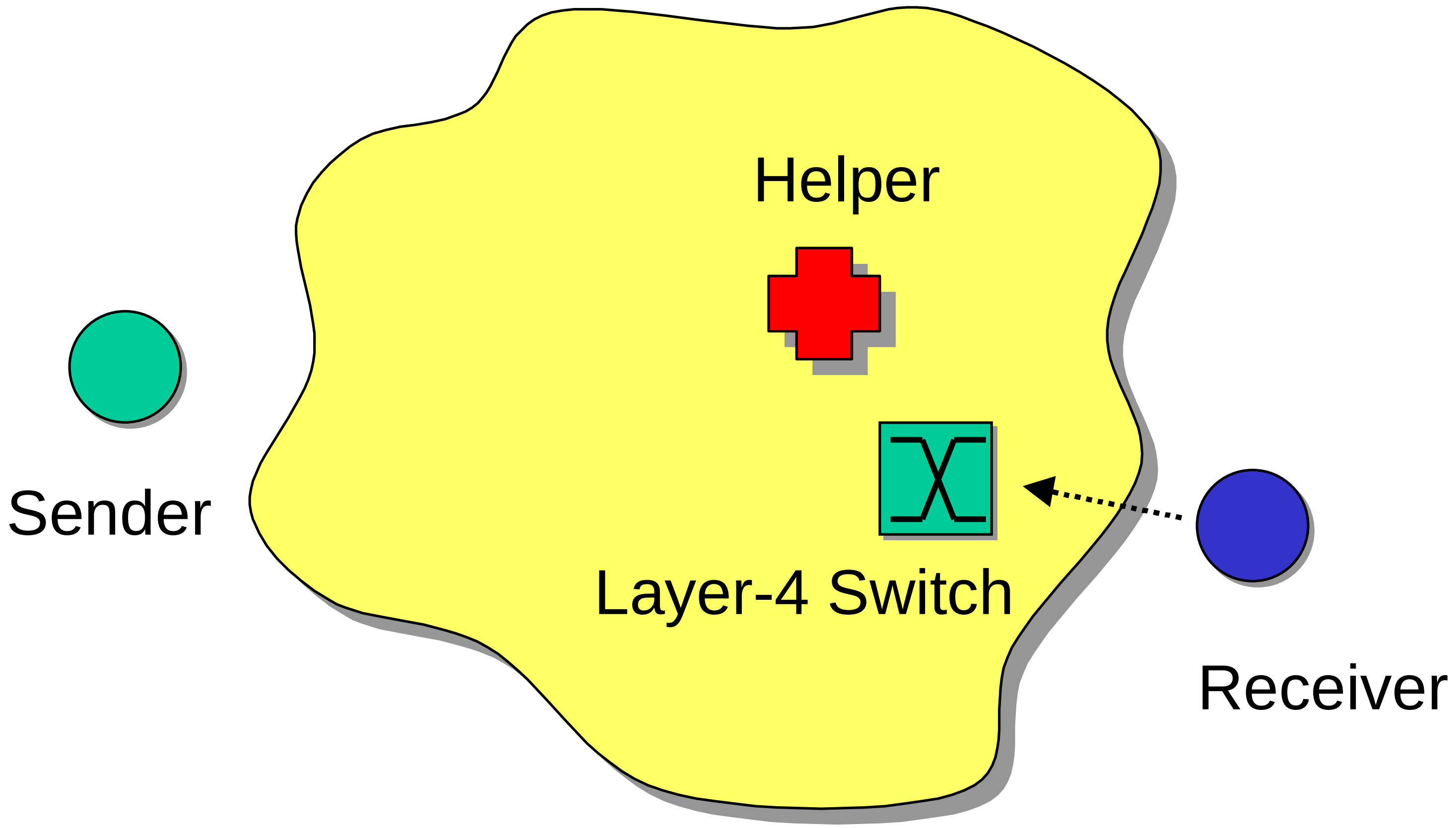
New Streaming Architecture



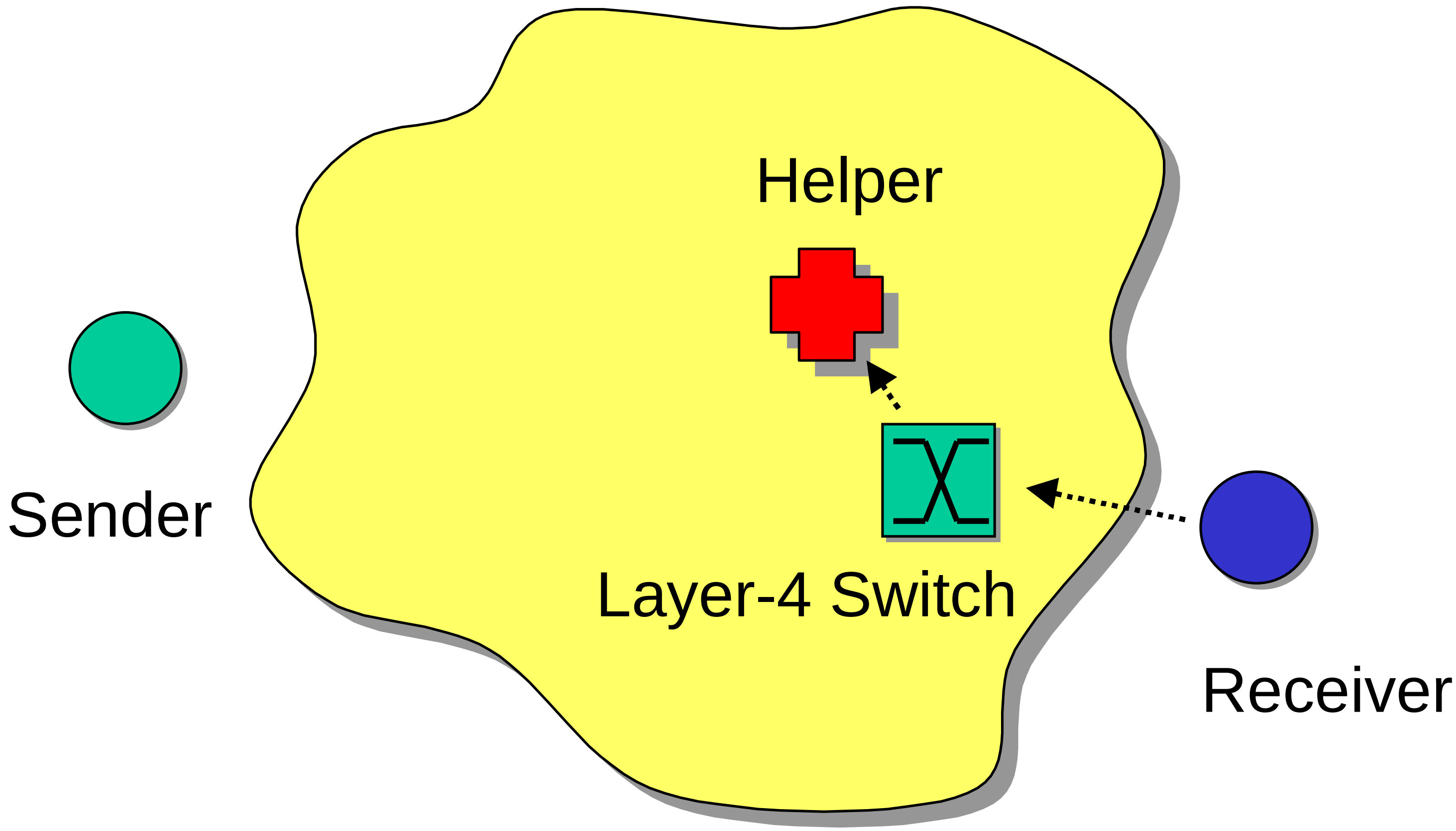
Transparent Operation



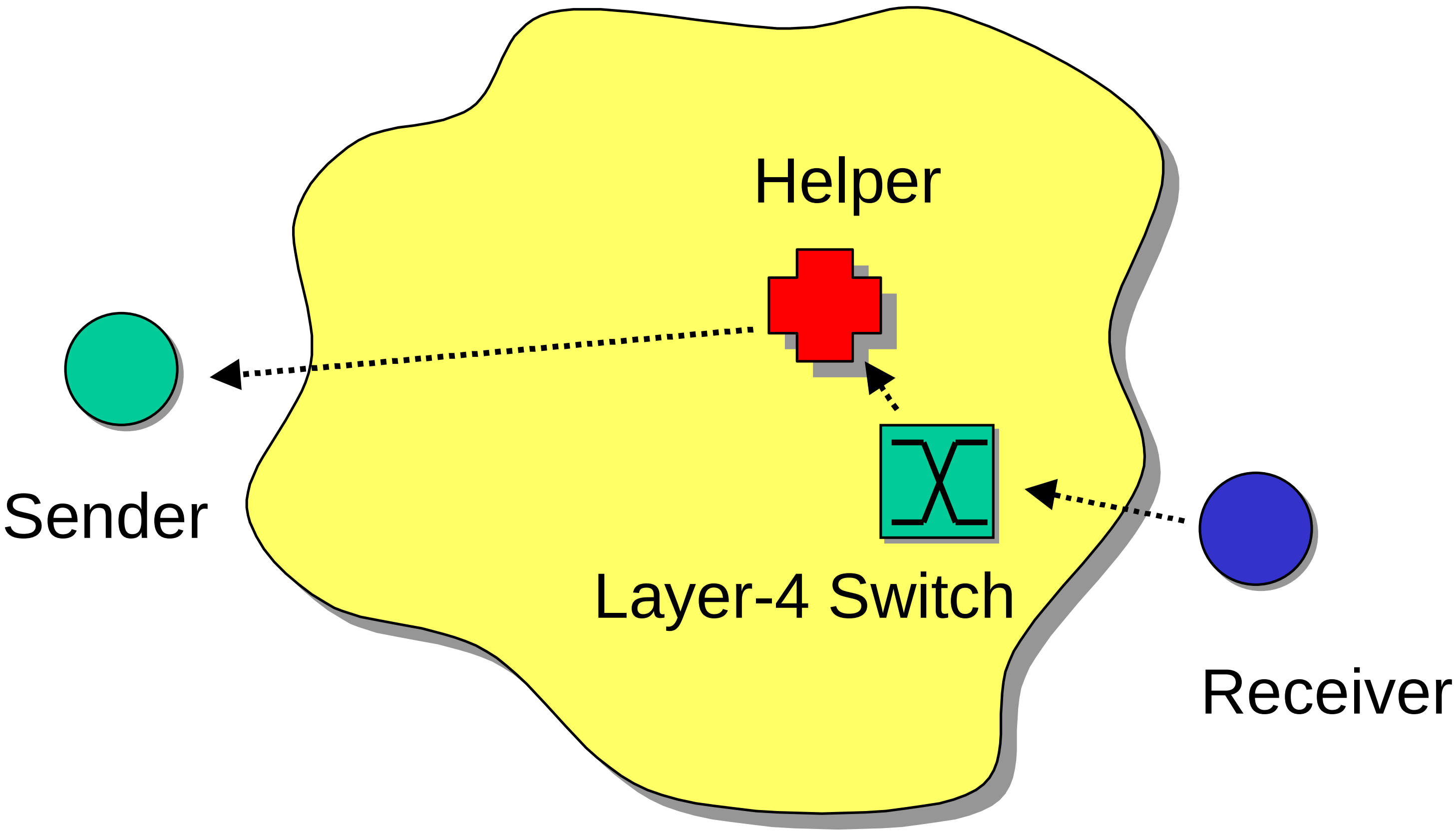
Transparent Operation



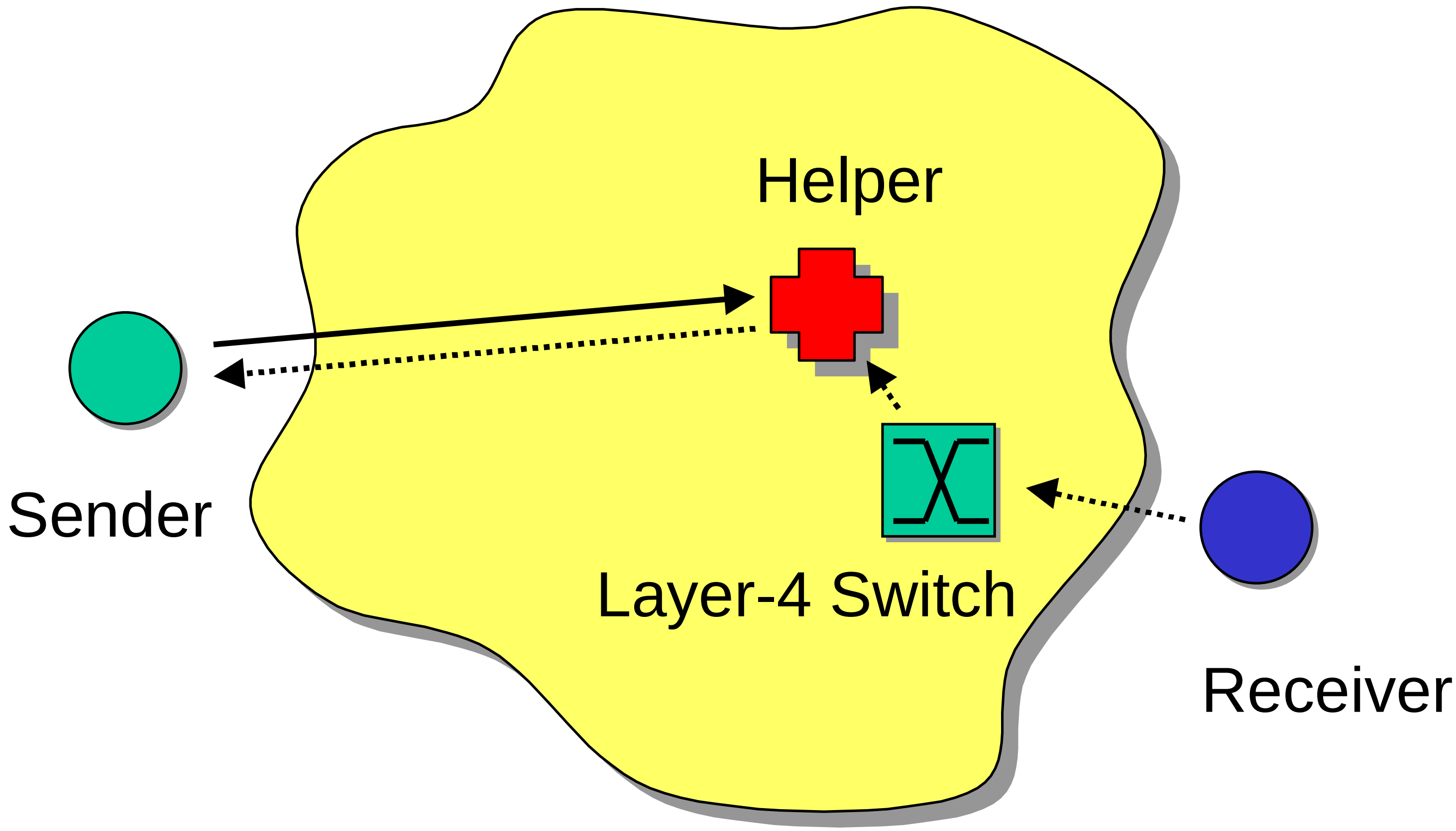
Transparent Operation



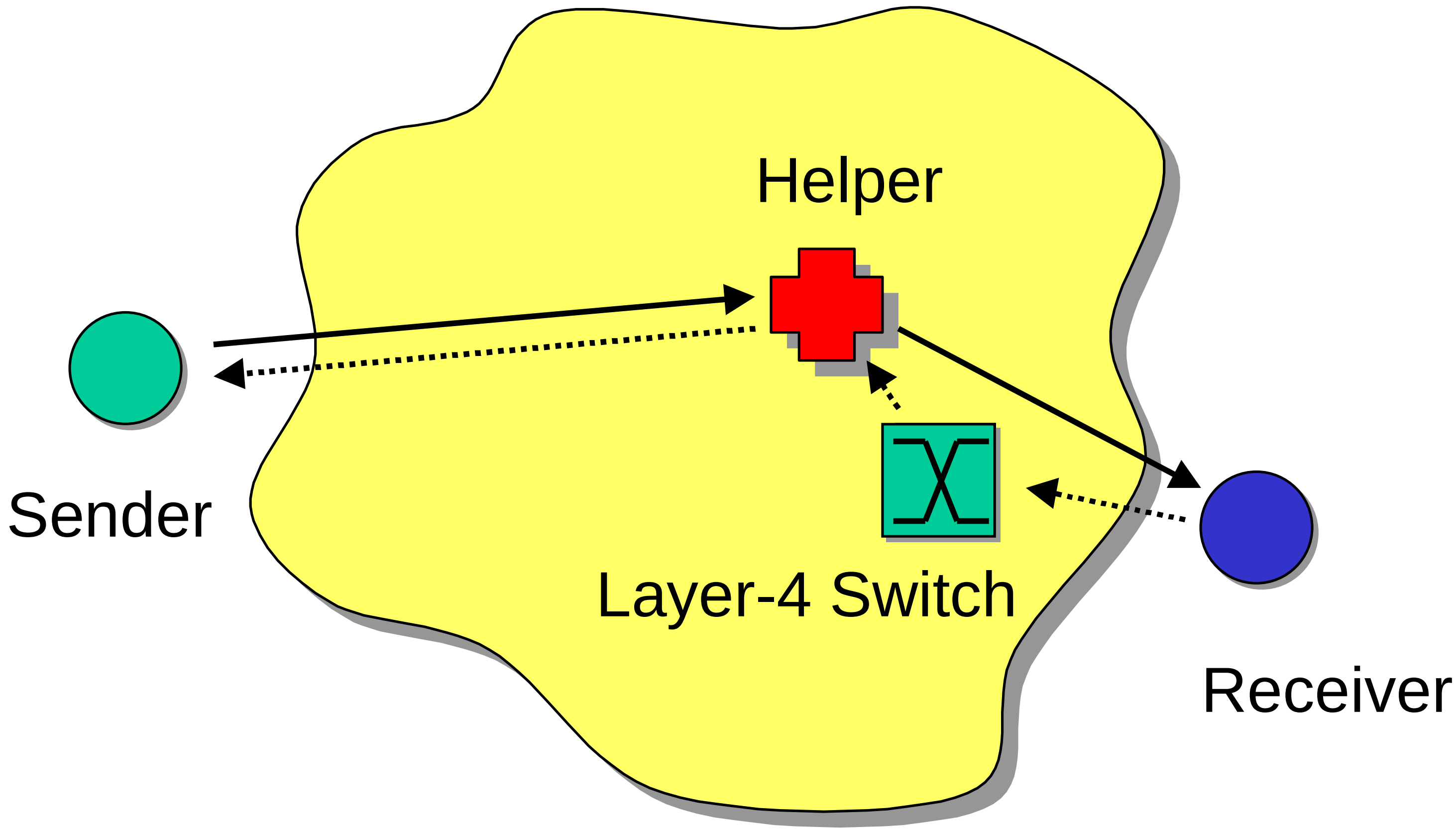
Transparent Operation



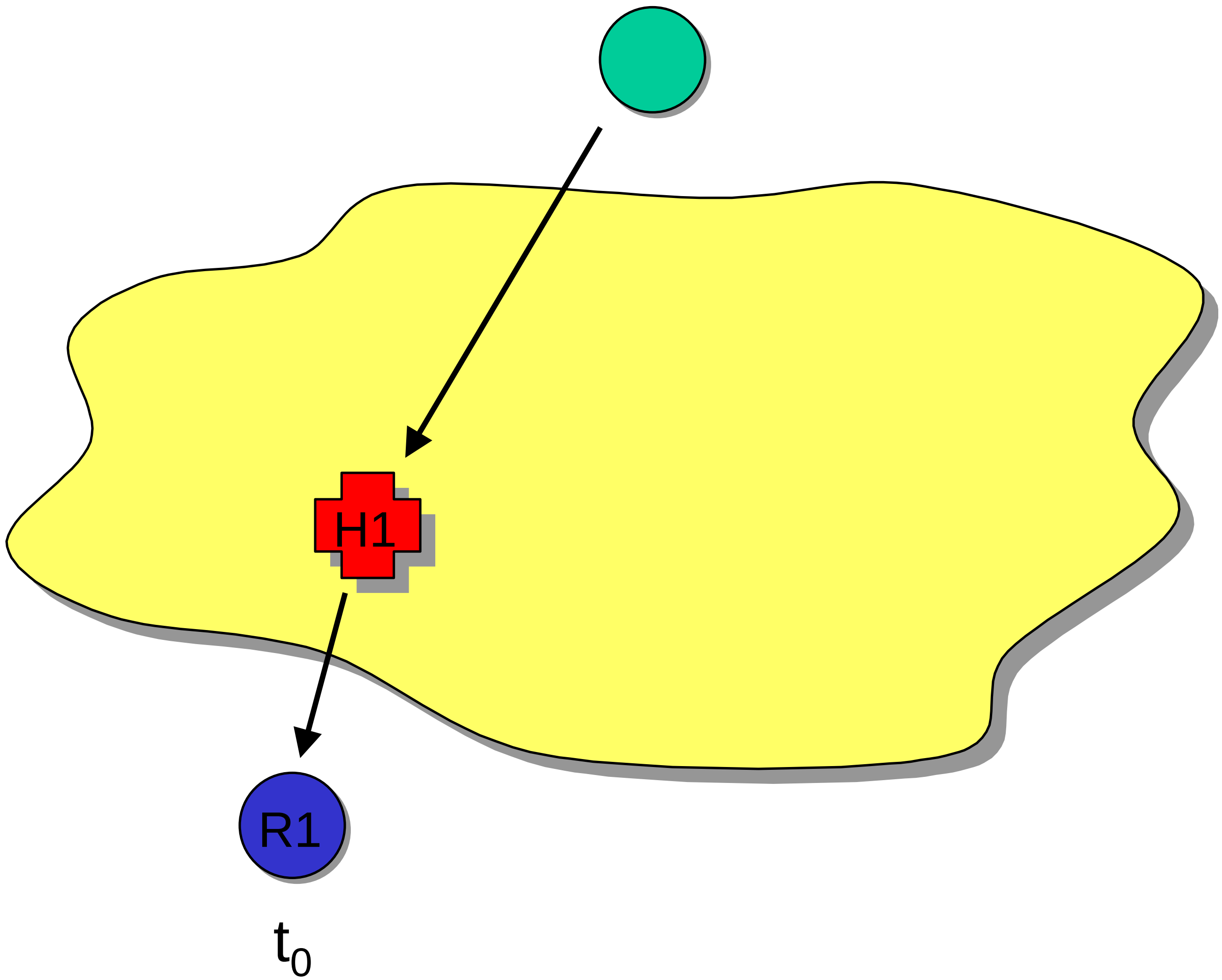
Transparent Operation



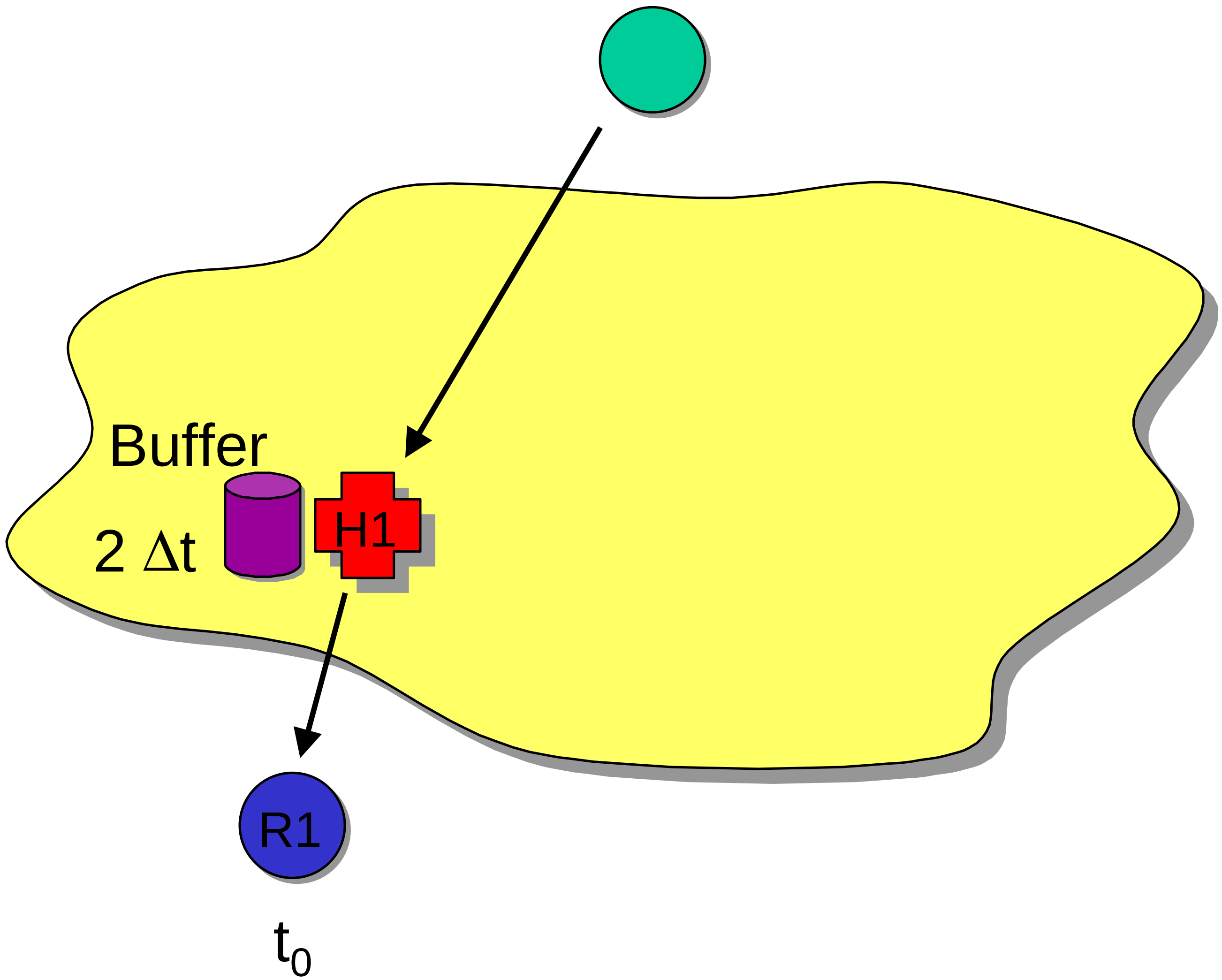
Transparent Operation



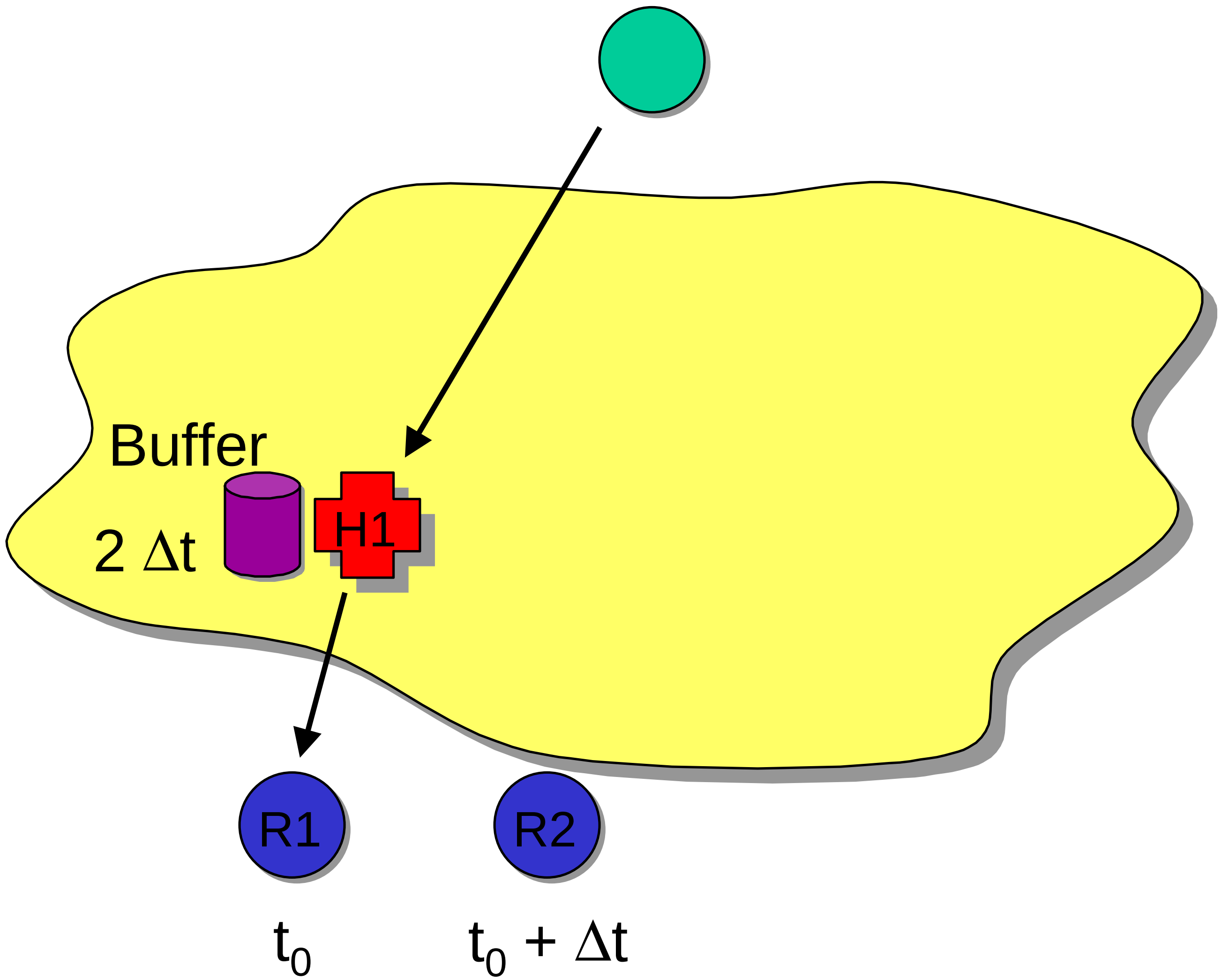
Helper Mesh Formation



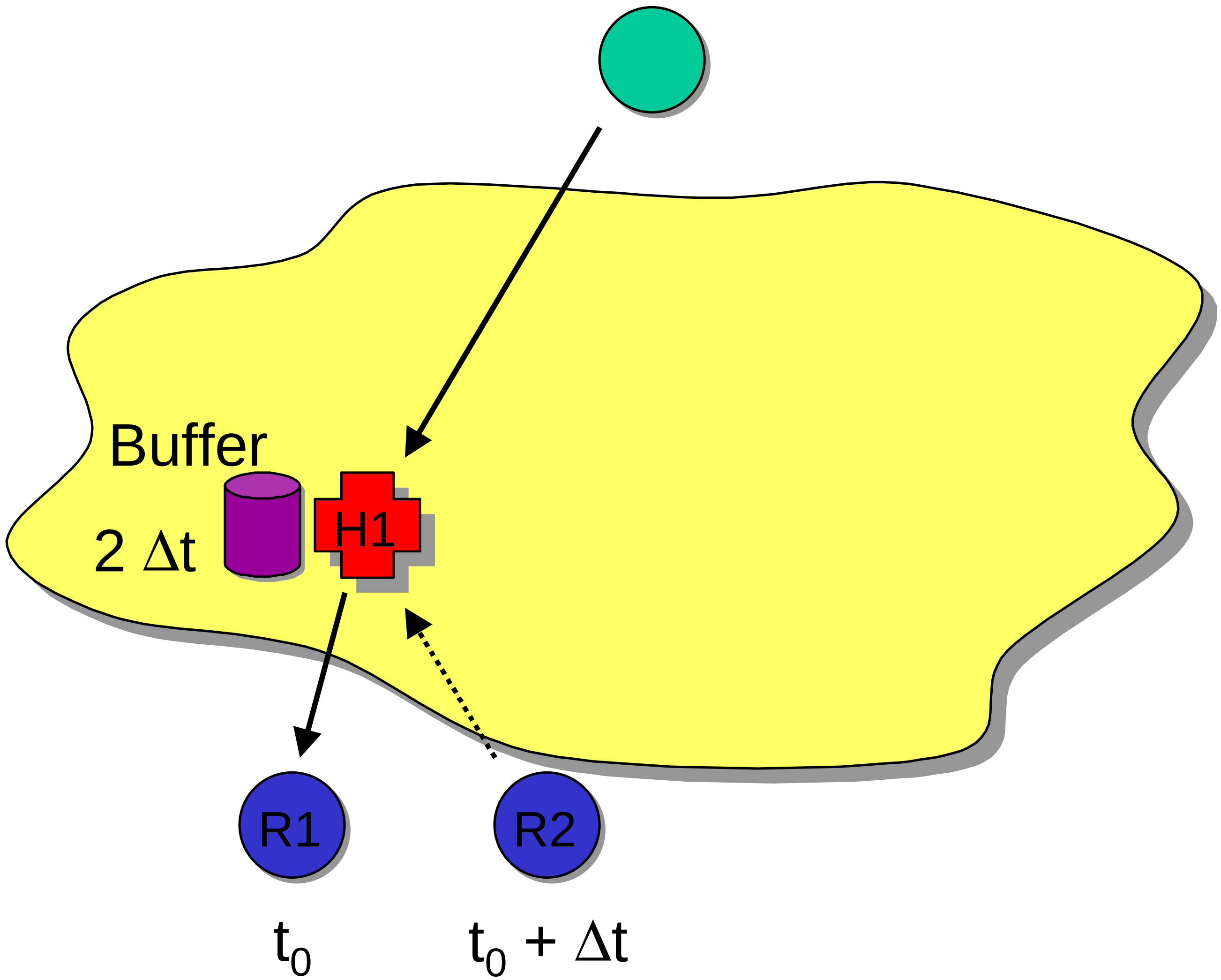
Helper Mesh Formation



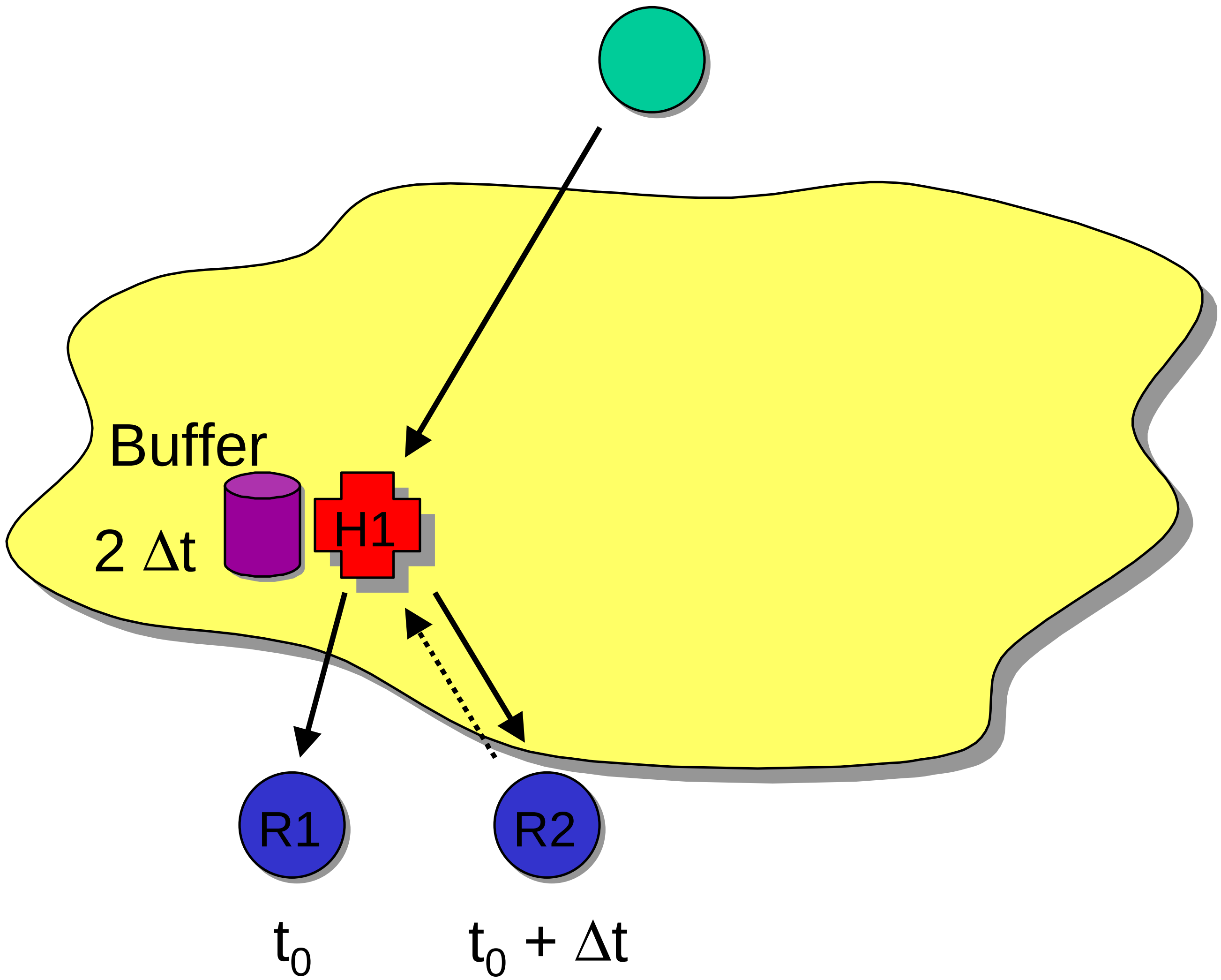
Helper Mesh Formation



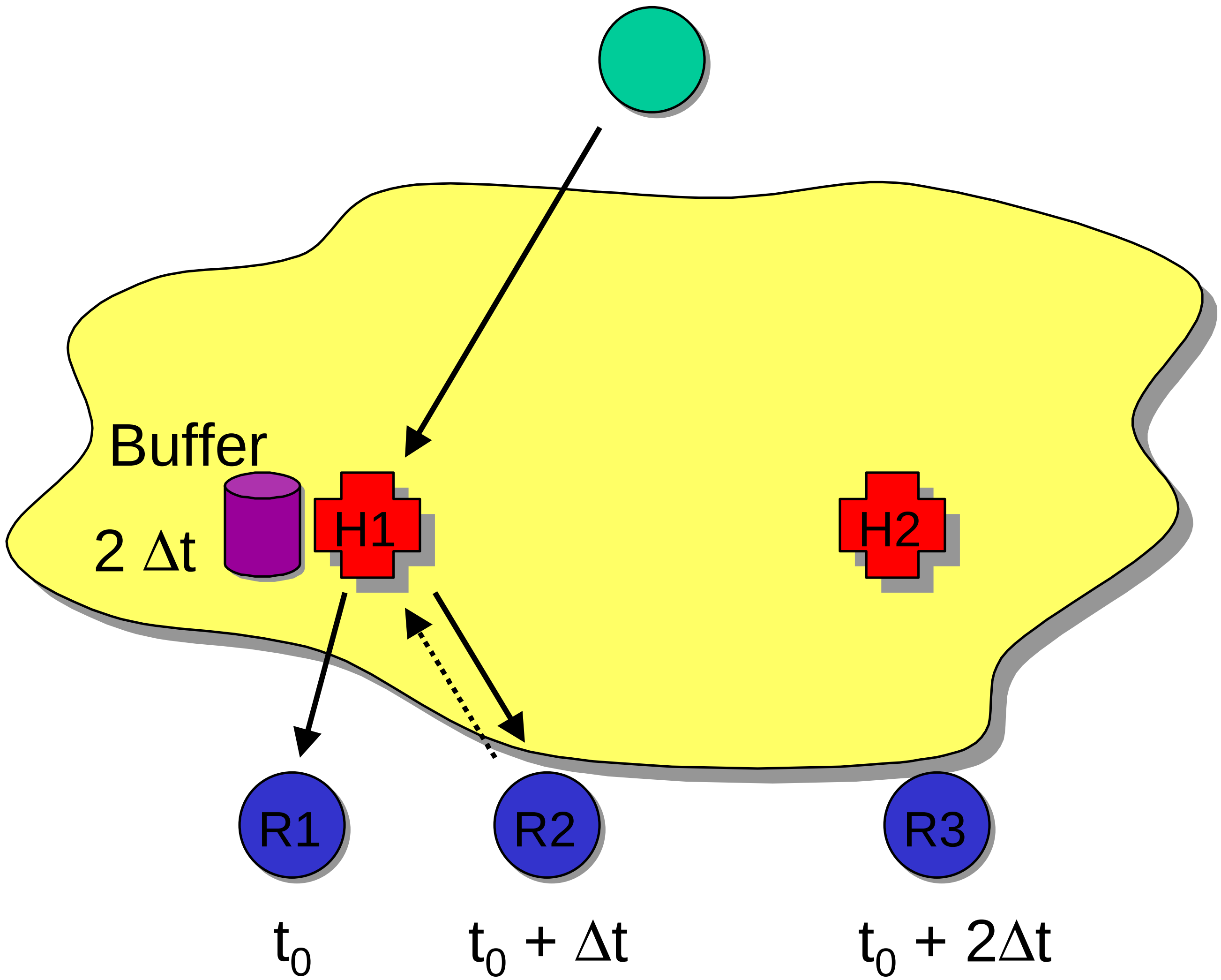
Helper Mesh Formation



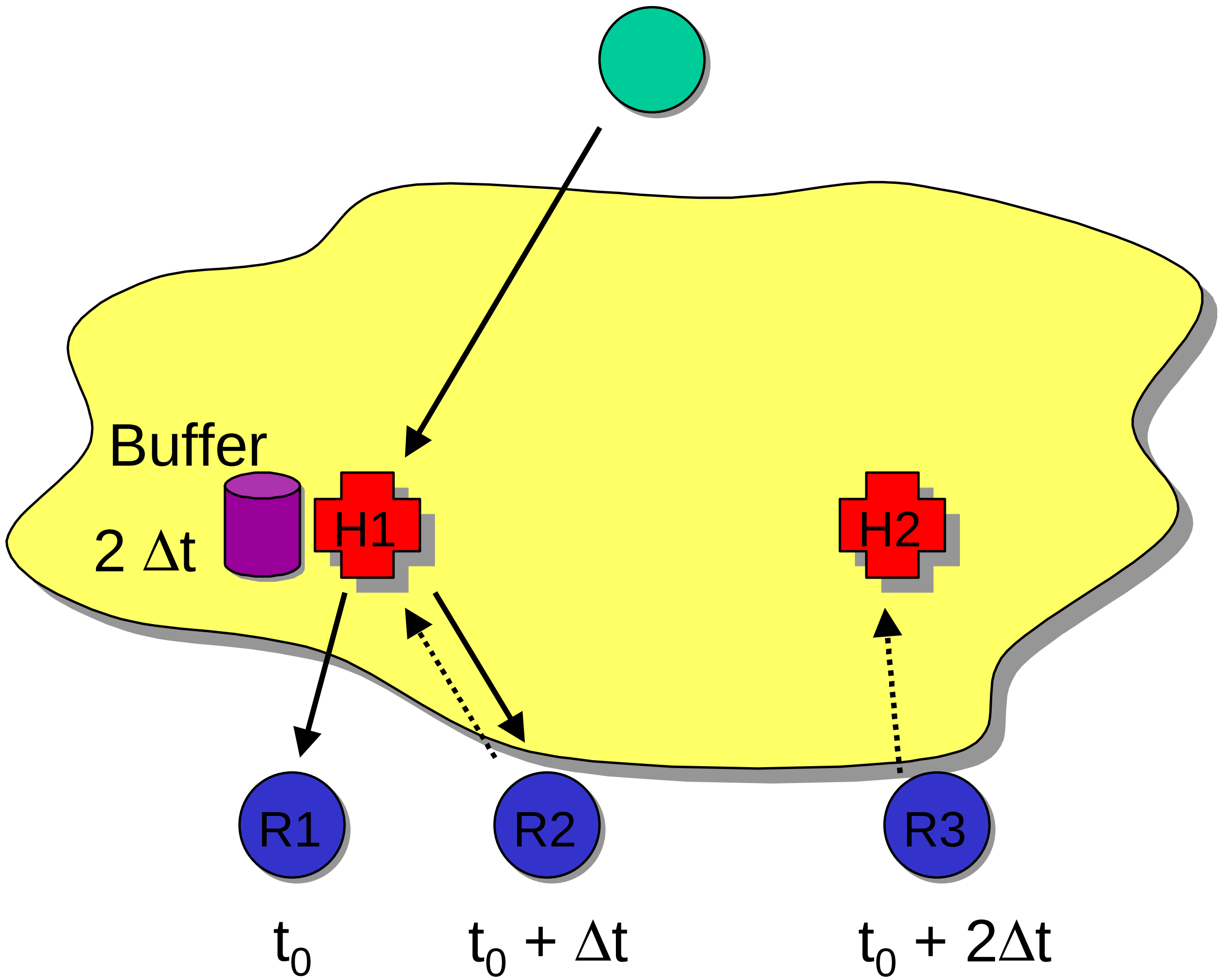
Helper Mesh Formation



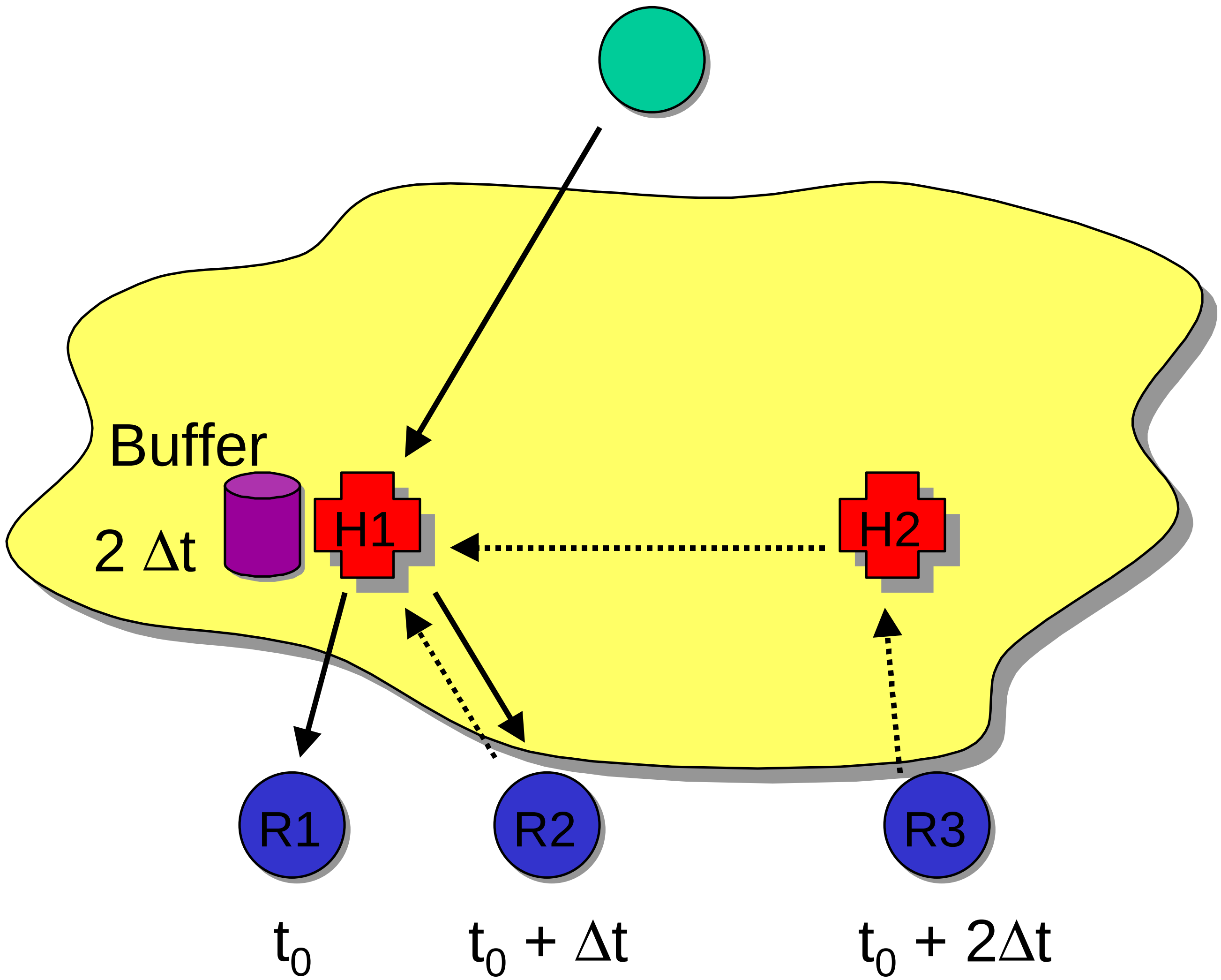
Helper Mesh Formation



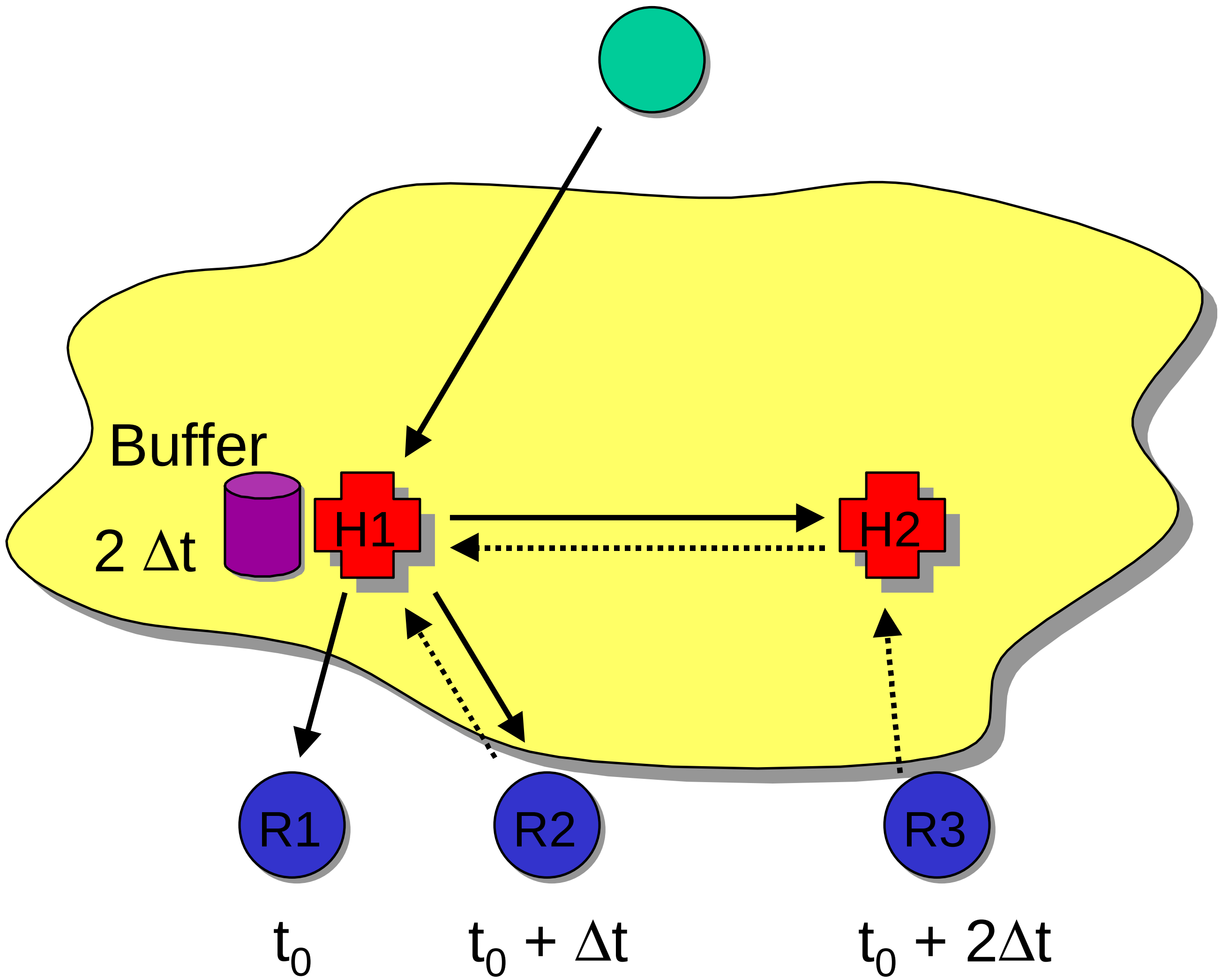
Helper Mesh Formation



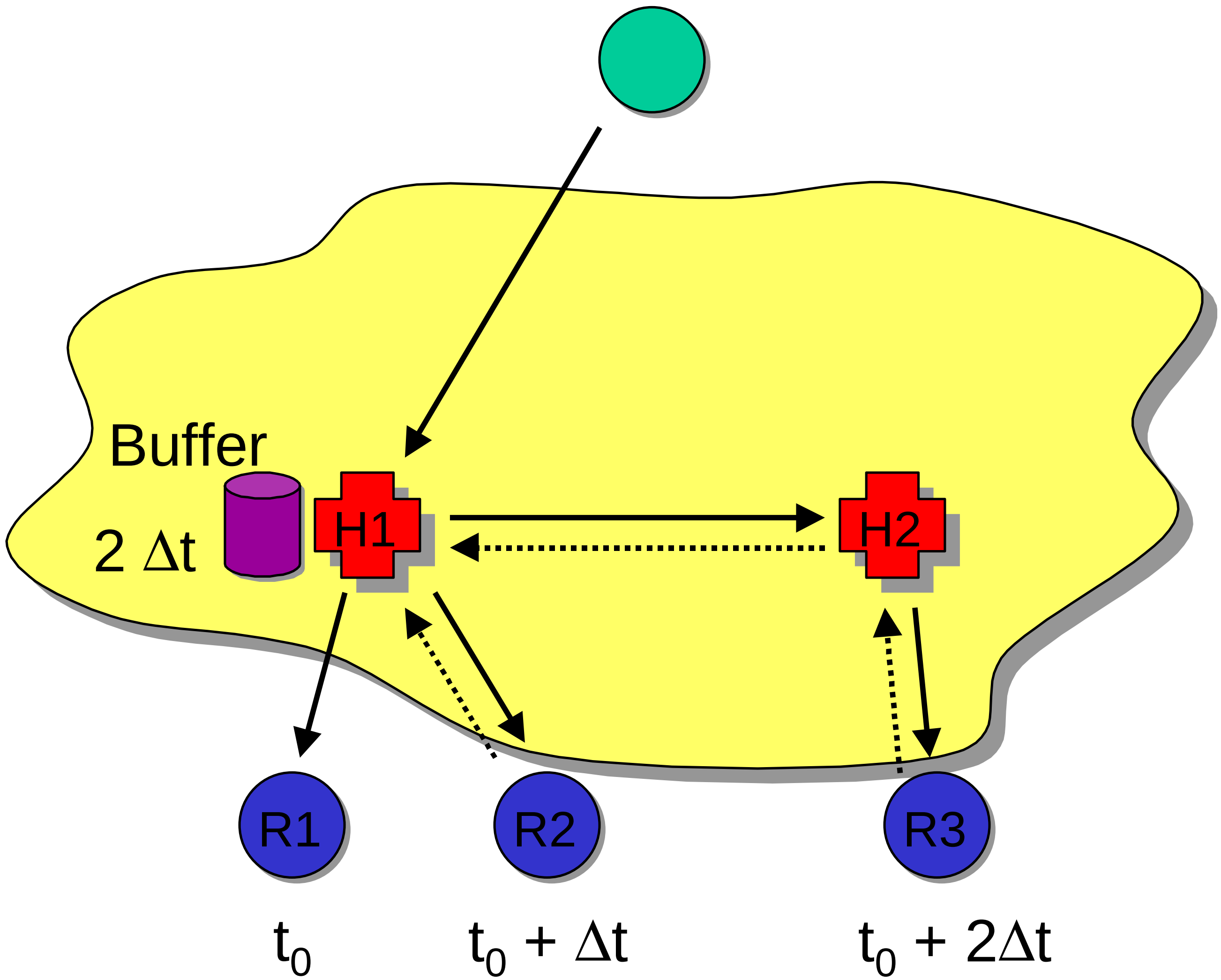
Helper Mesh Formation



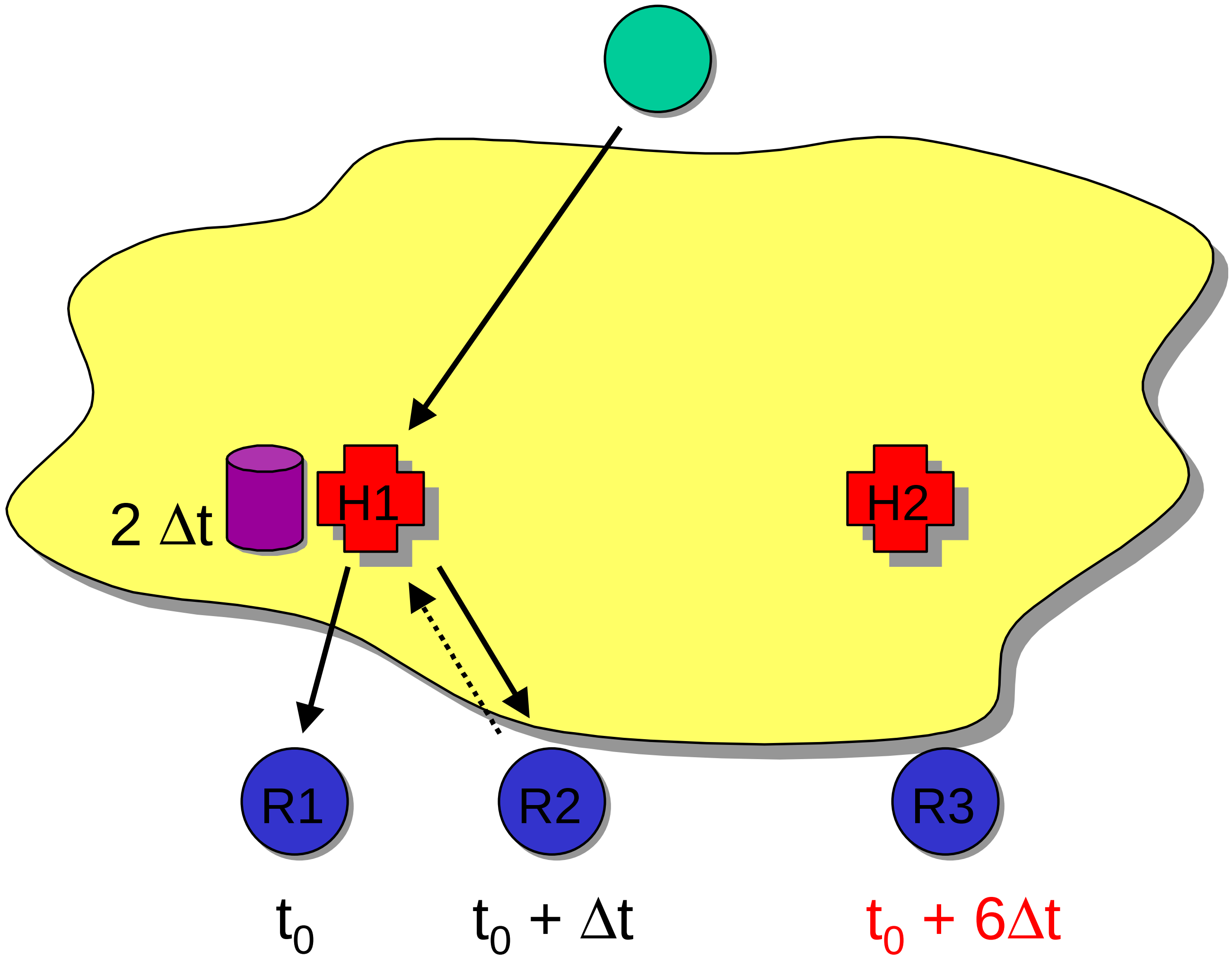
Helper Mesh Formation



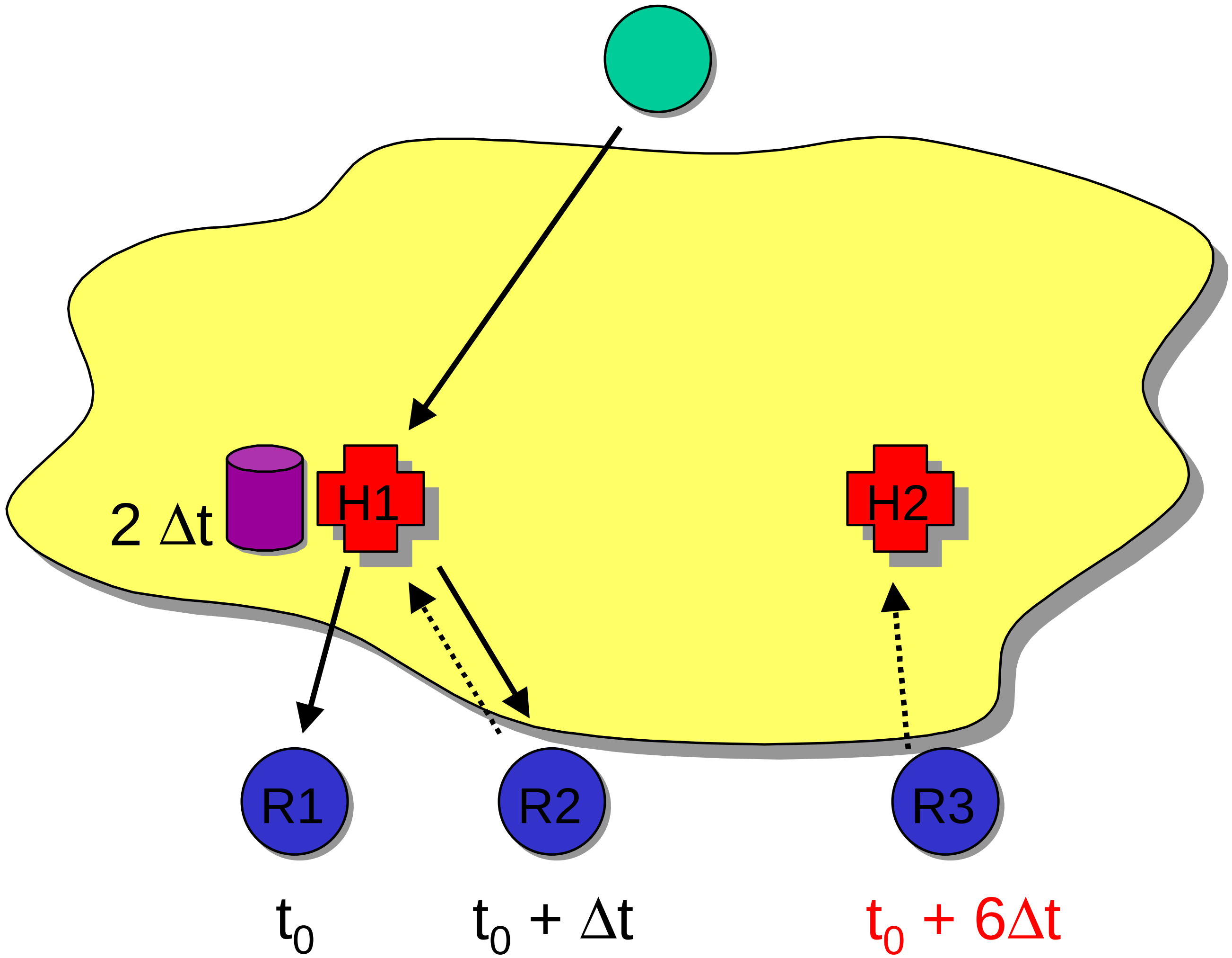
Helper Mesh Formation



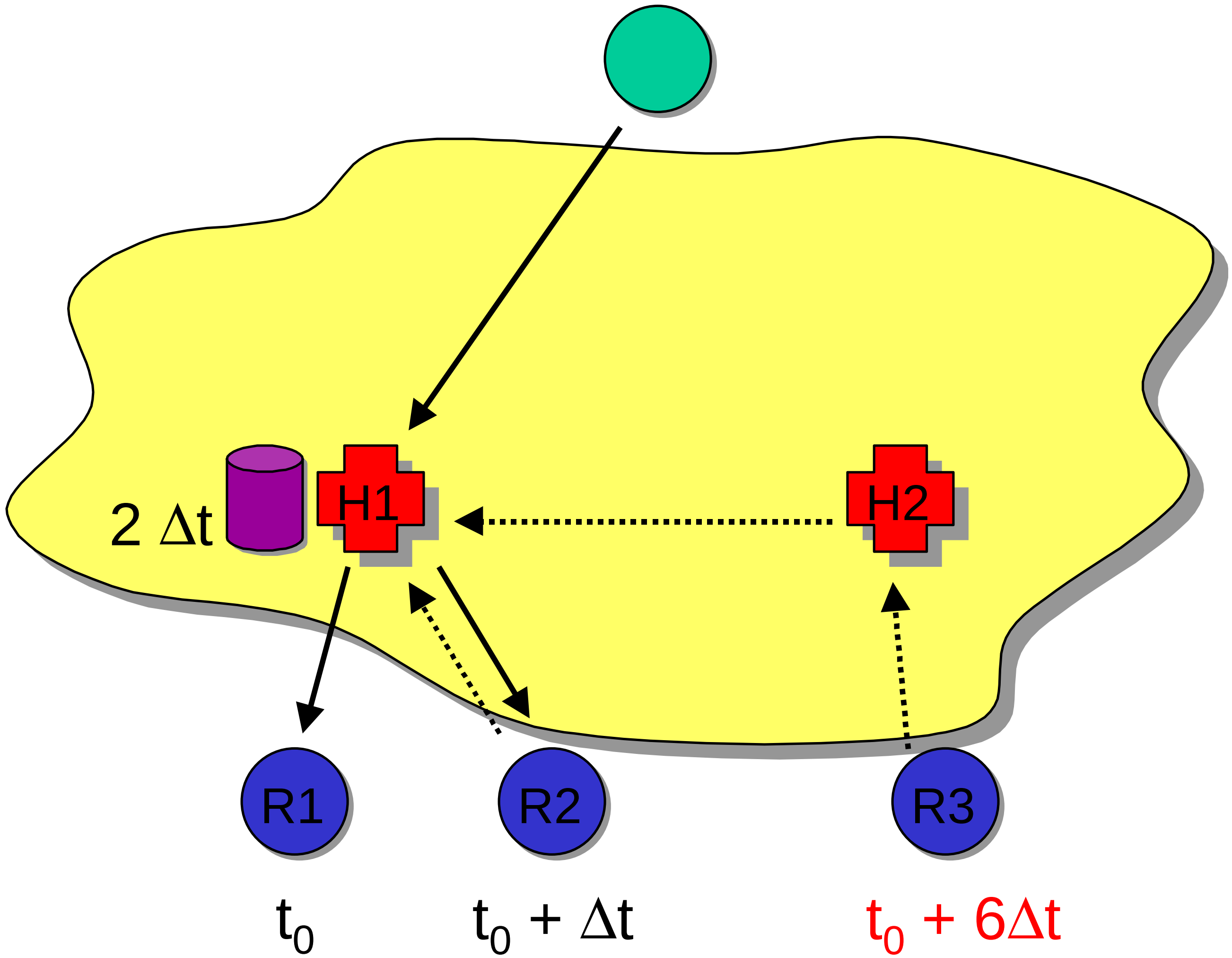
Using Buffer in the General Case



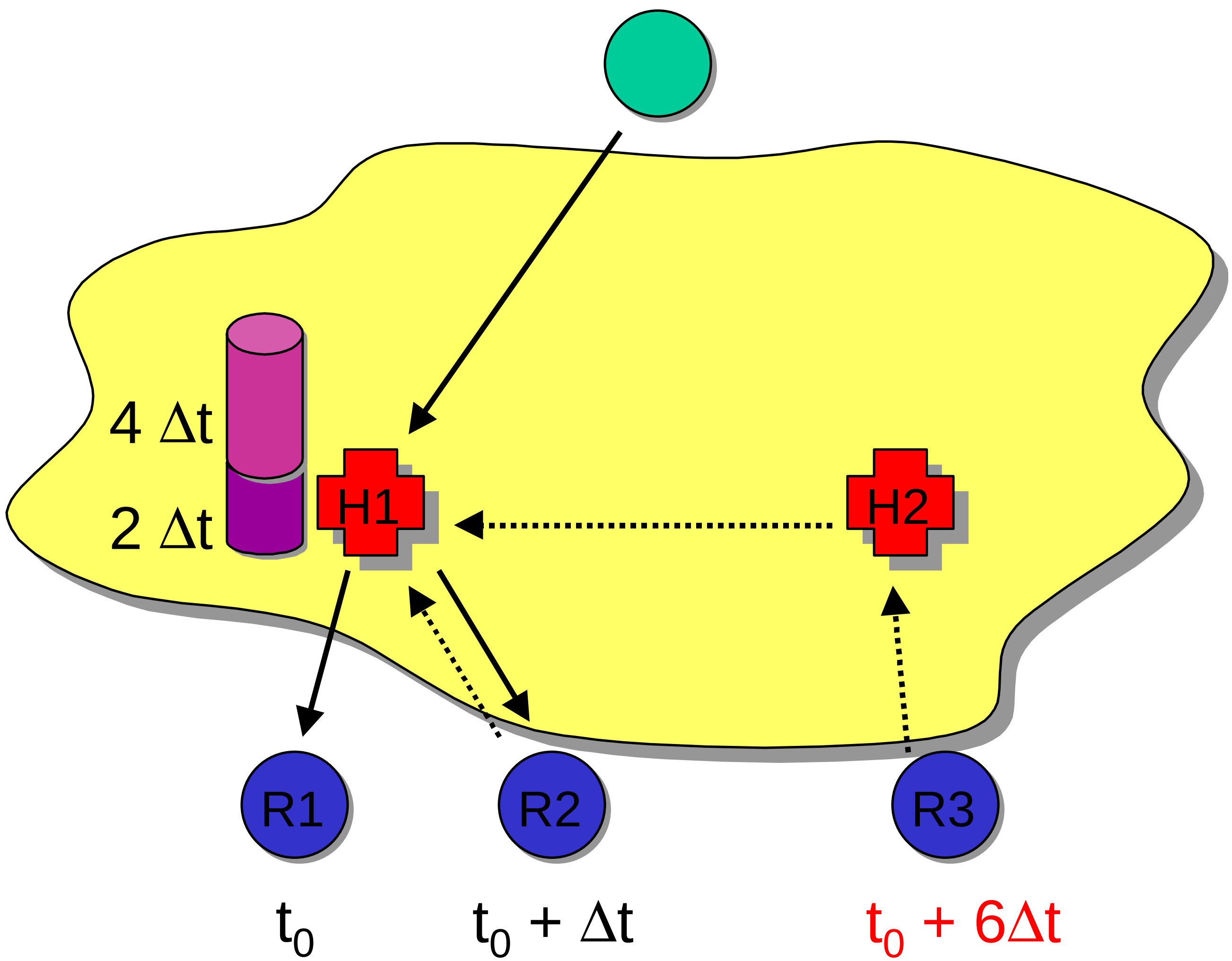
Using Buffer in the General Case



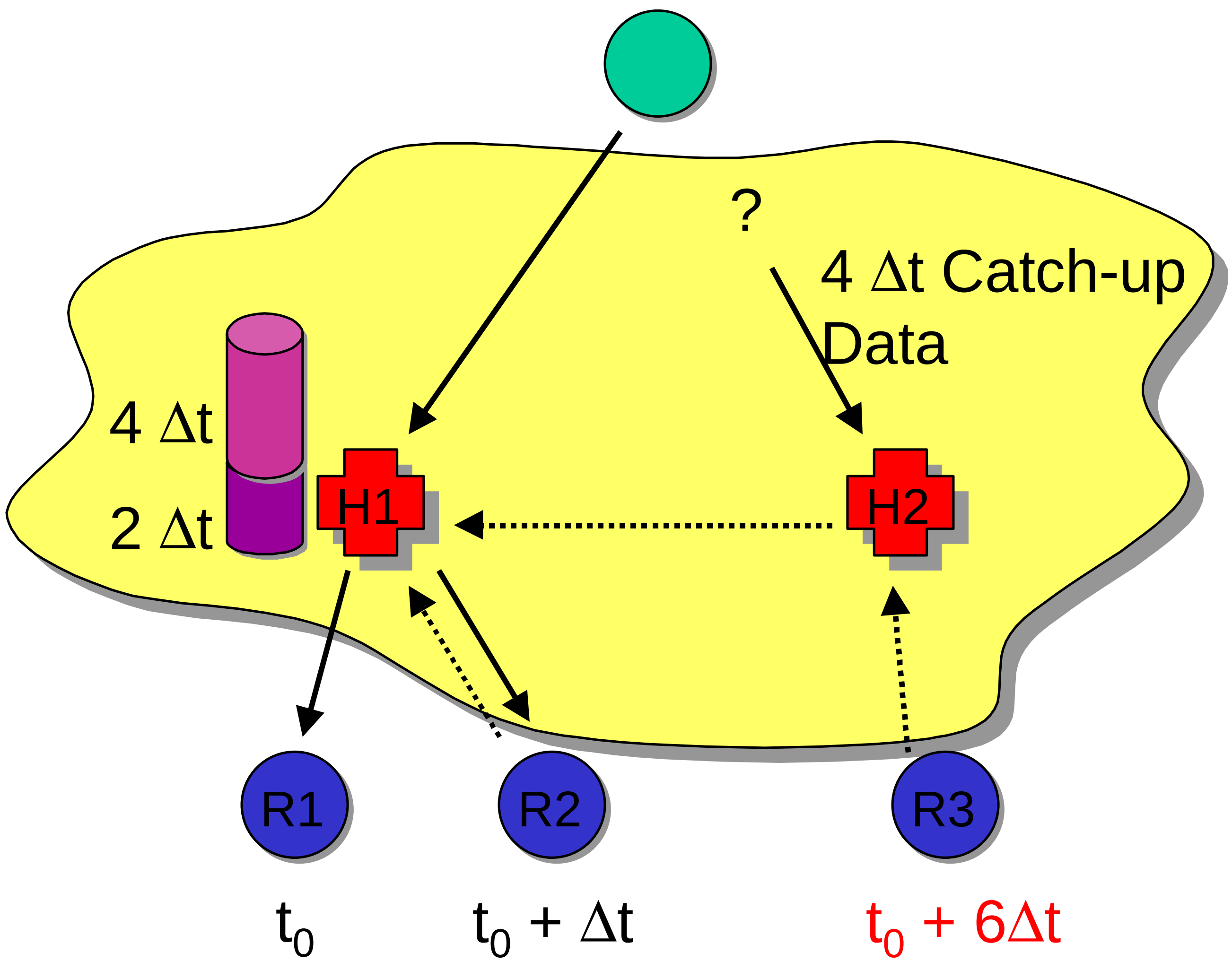
Using Buffer in the General Case



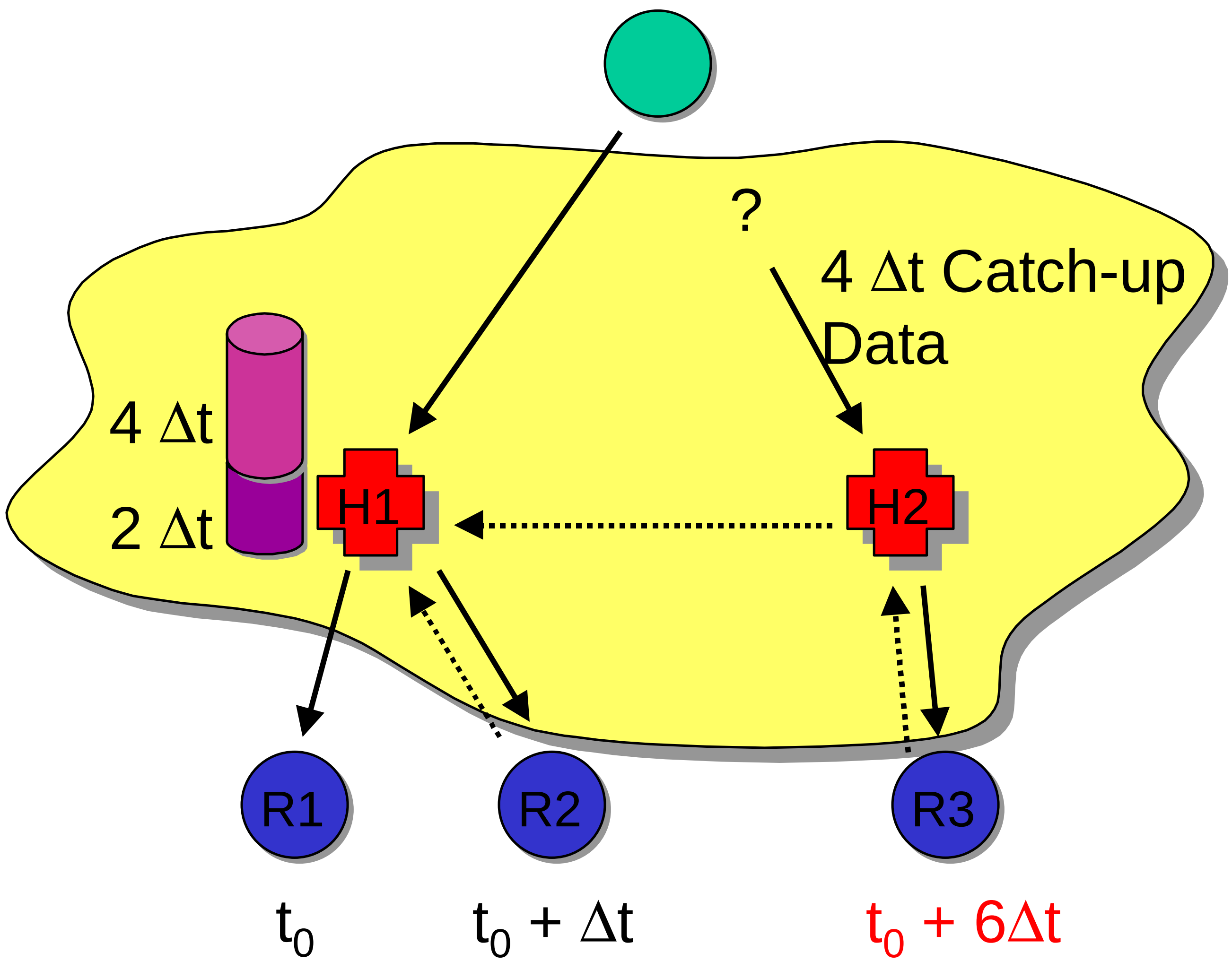
Using Buffer in the General Case



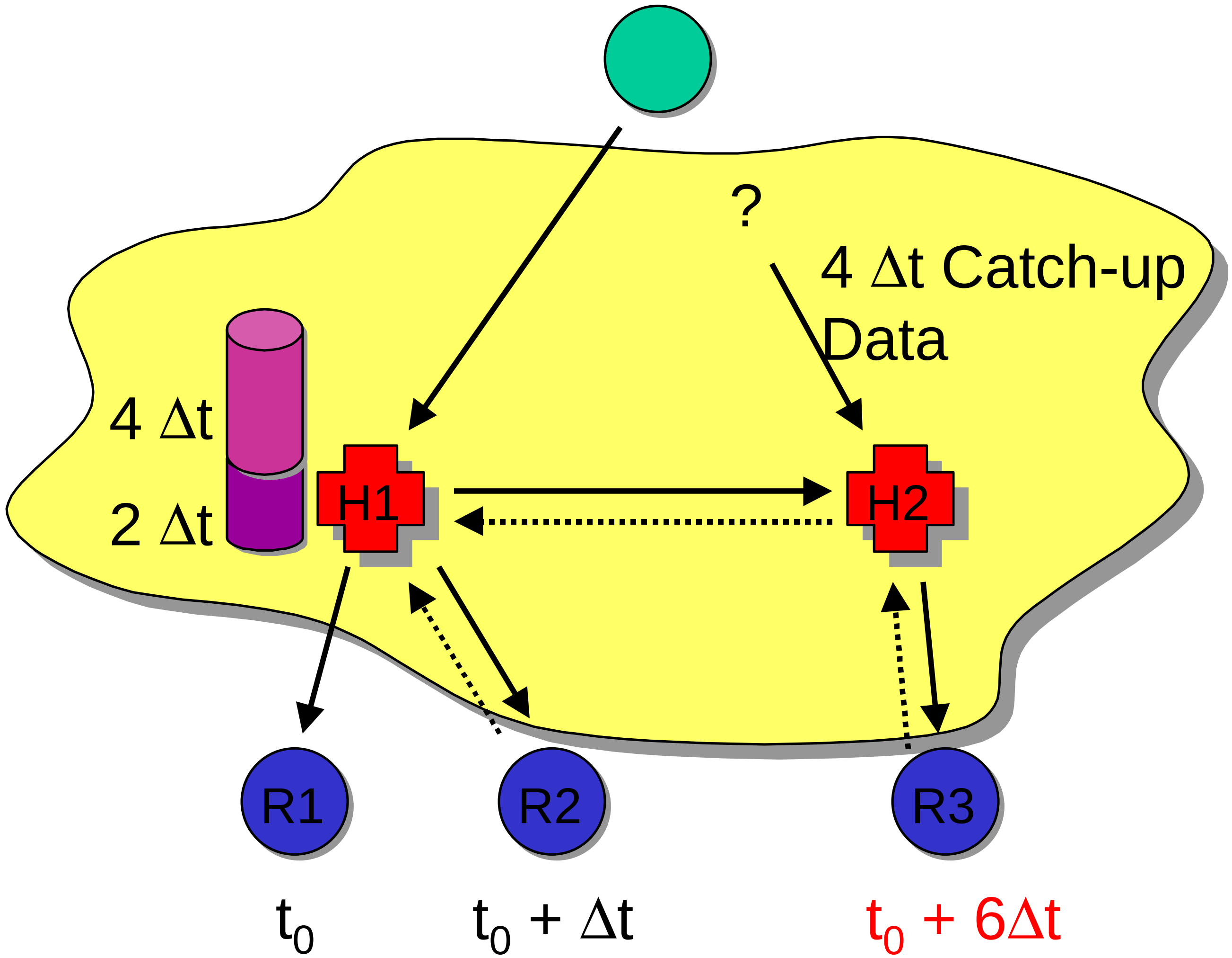
Using Buffer in the General Case



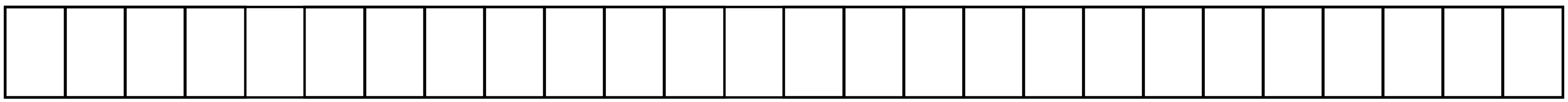
Using Buffer in the General Case

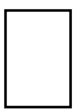


Using Buffer in the General Case



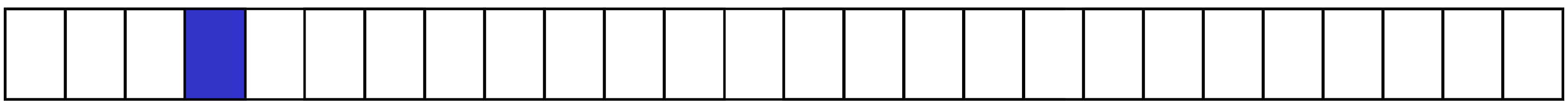
Defining Parameters Using Segments



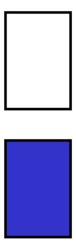


Segment

Defining Parameters Using Segments



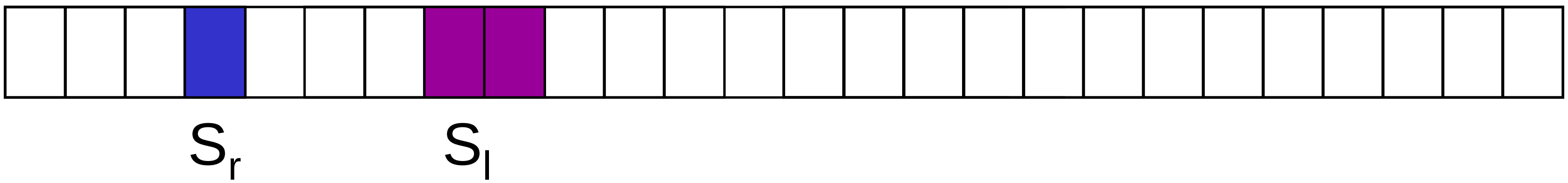
S_r

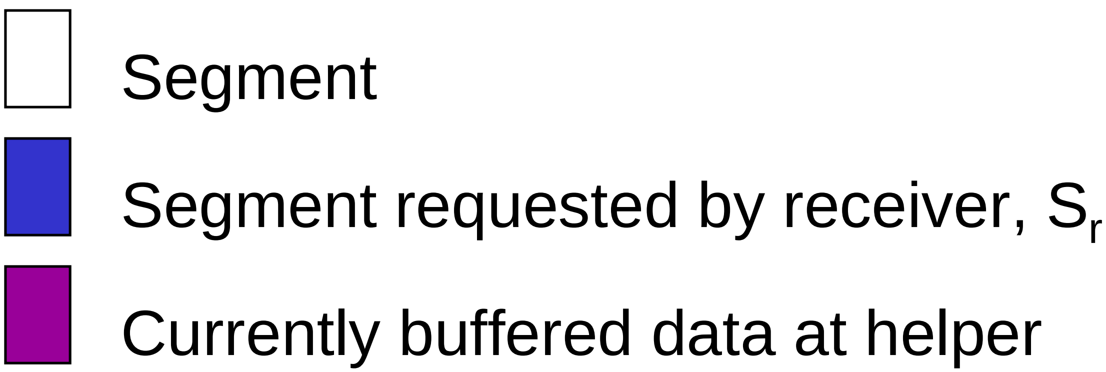


Segment

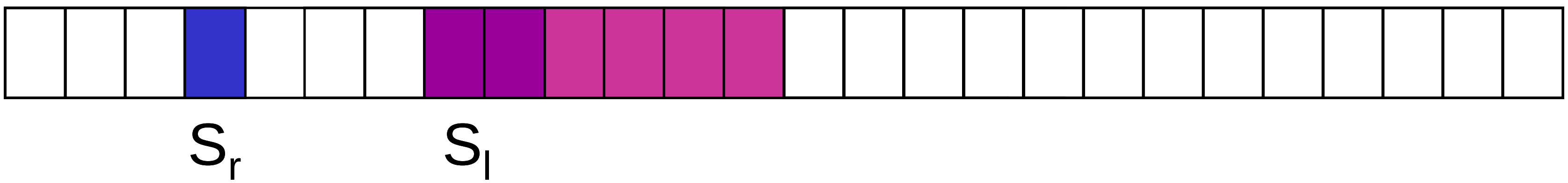
Segment requested by receiver, S_r

Defining Parameters Using Segments



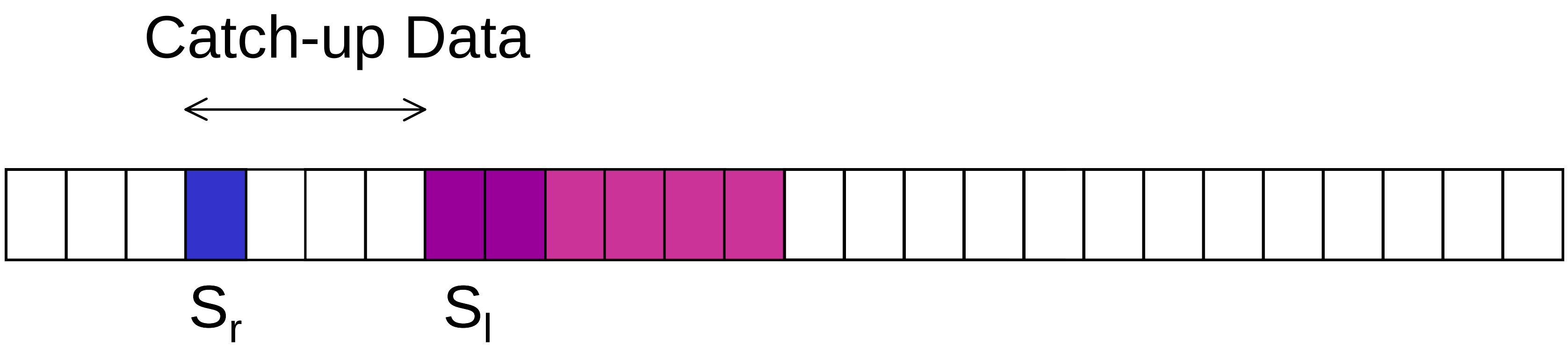


Defining Parameters Using Segments



- Segment
- Segment requested by receiver, S_r
- Currently buffered data at helper
- Additional buffer required to bridge the gap, S_l - S_r

Defining Parameters Using Segments



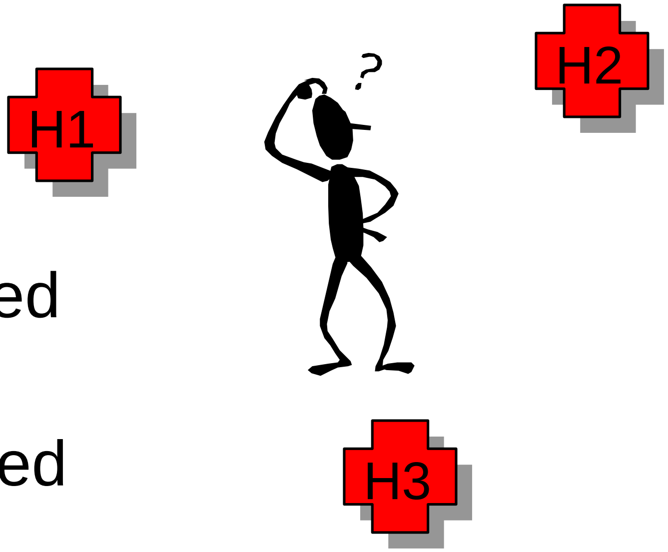
Segment

Segment requested by receiver, S_r

Currently buffered data at helper

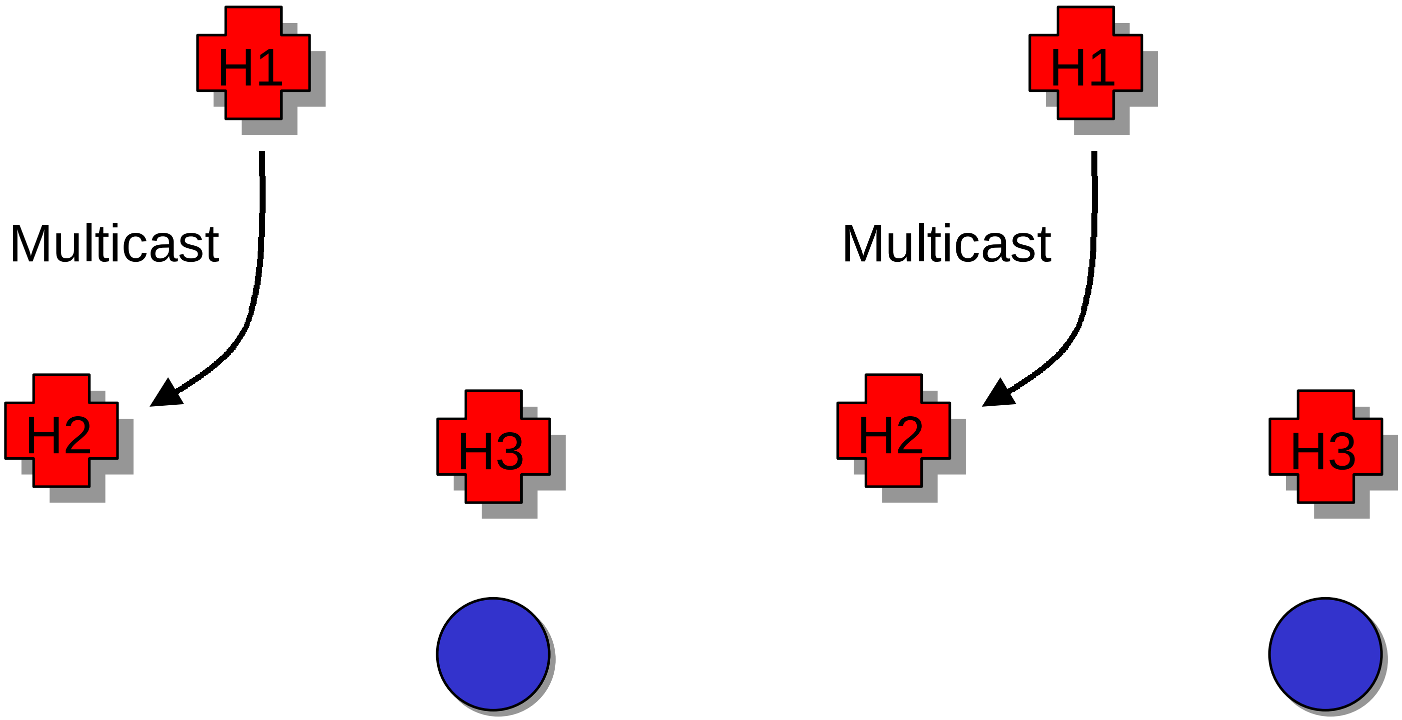
Additional buffer required to bridge the gap, S_l - S_r

Helper Selection Considerations

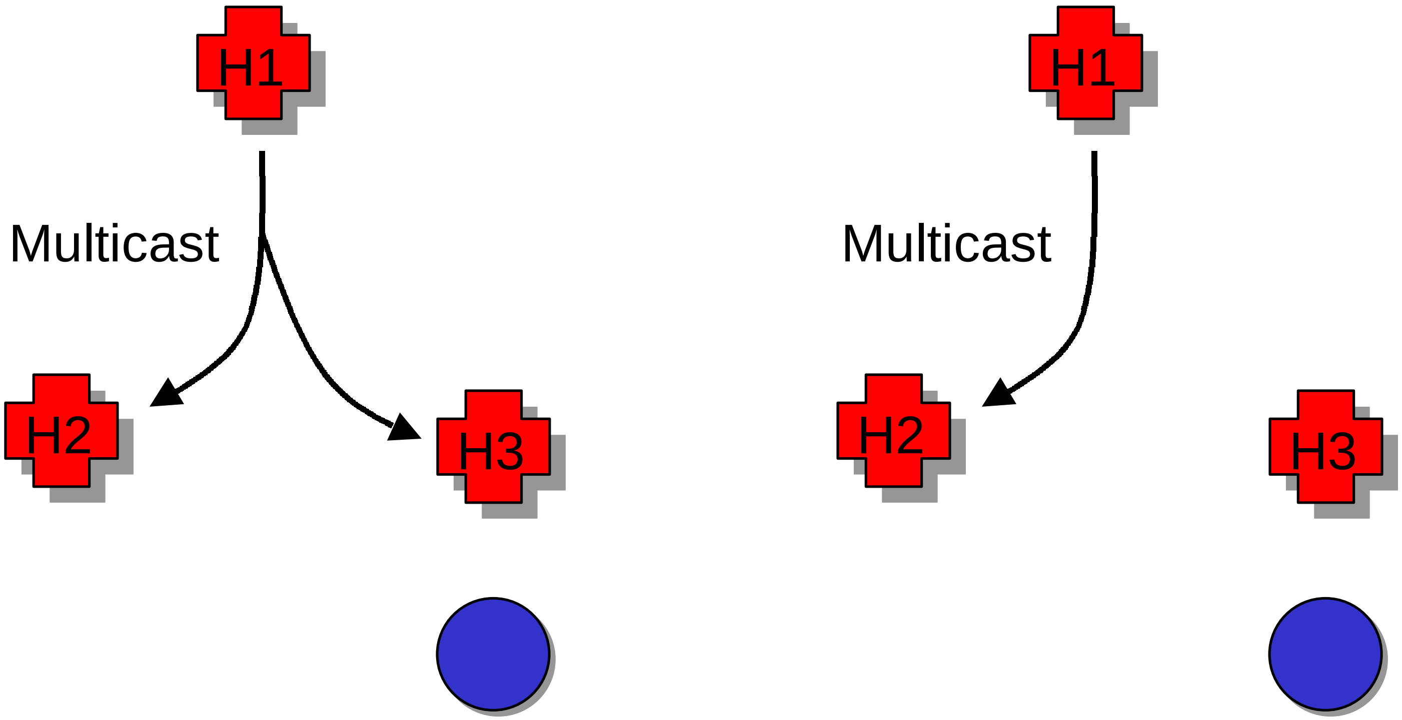


- Additional buffer space required
  - Knowledge of temporal distance
- Additional network load incurred
  - Knowledge of network distance
- Additional system load incurred
  - Bandwidth and buffer consumption
  - For load balancing
- Data stream sharing or not

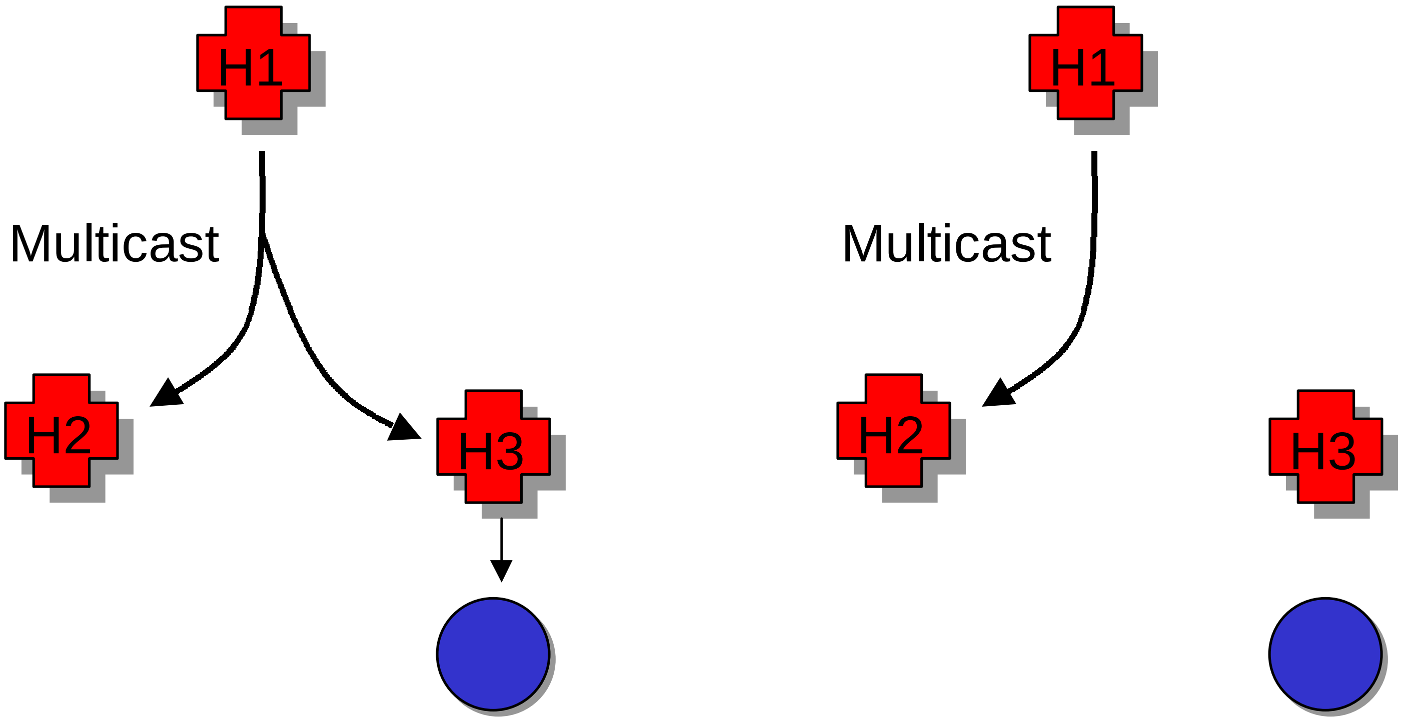
Data Stream Sharing or Not



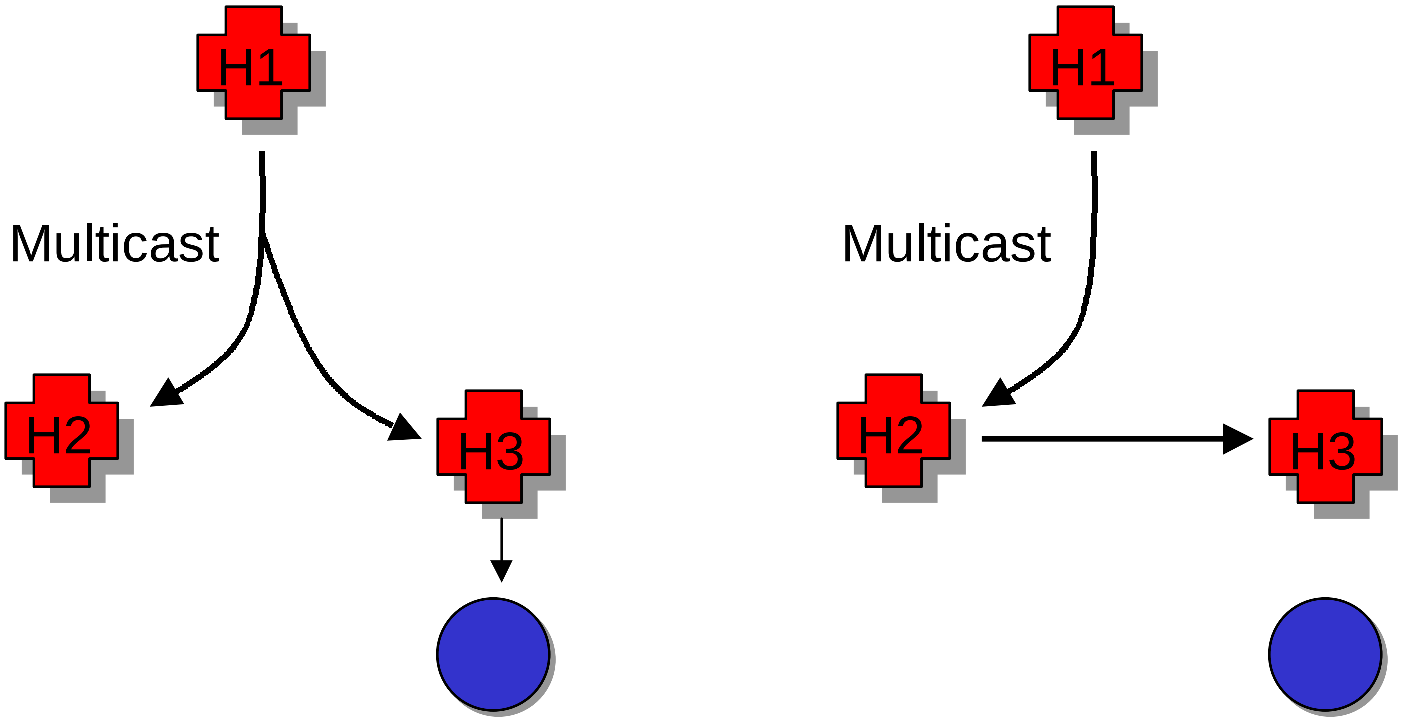
Data Stream Sharing or Not



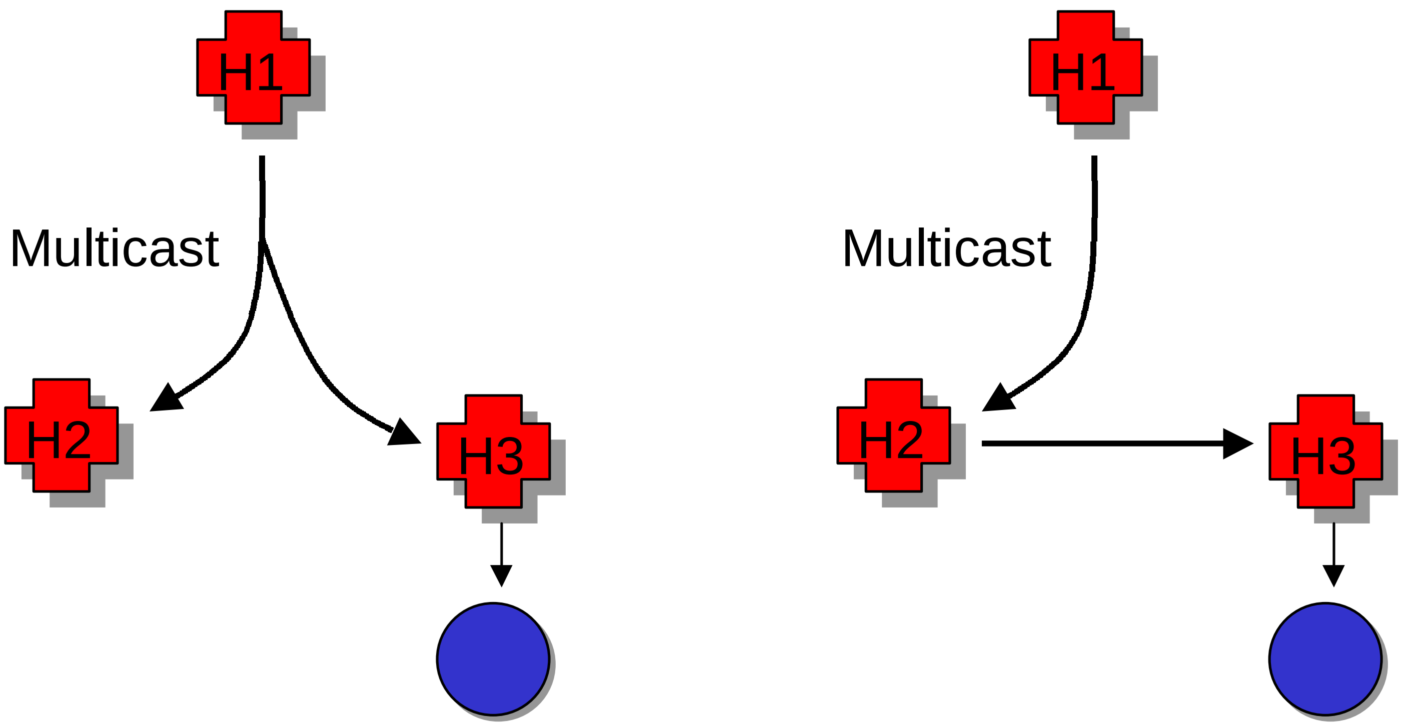
Data Stream Sharing or Not



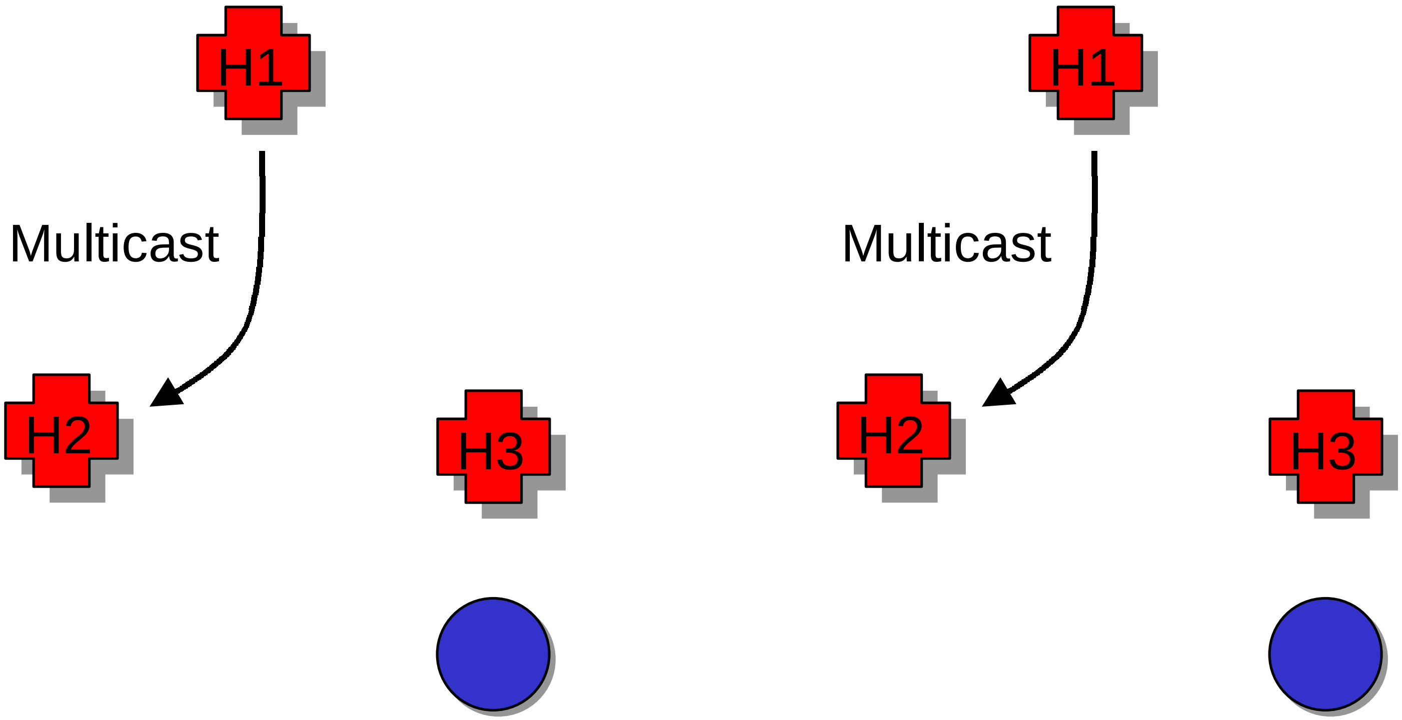
Data Stream Sharing or Not



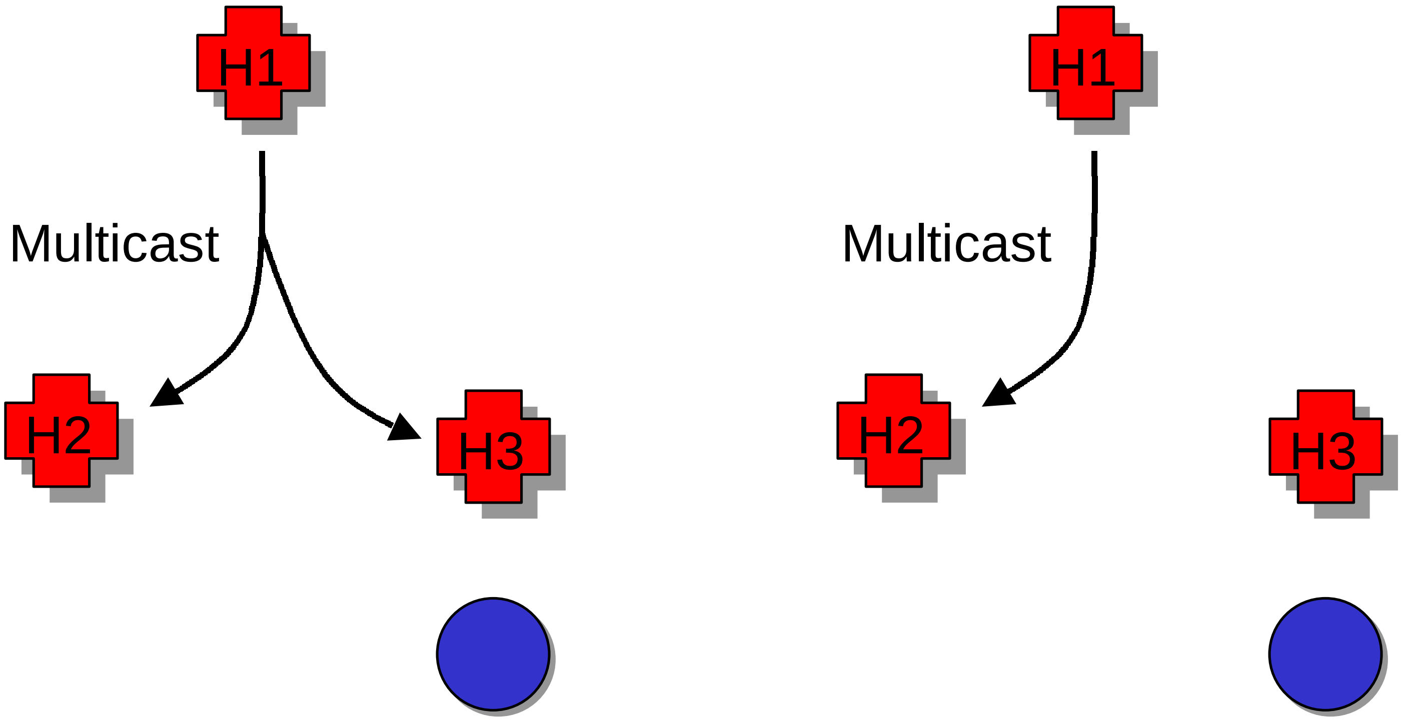
Data Stream Sharing or Not



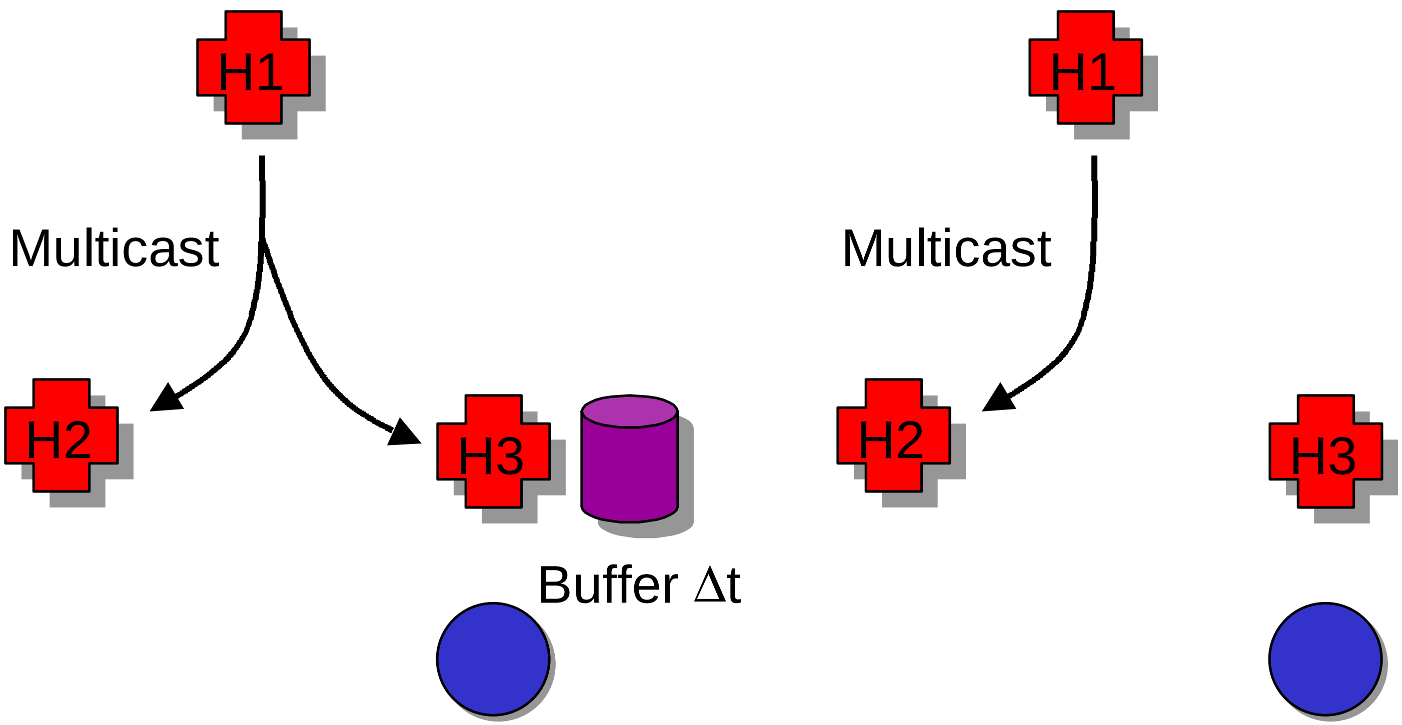
Data Stream Sharing or Not



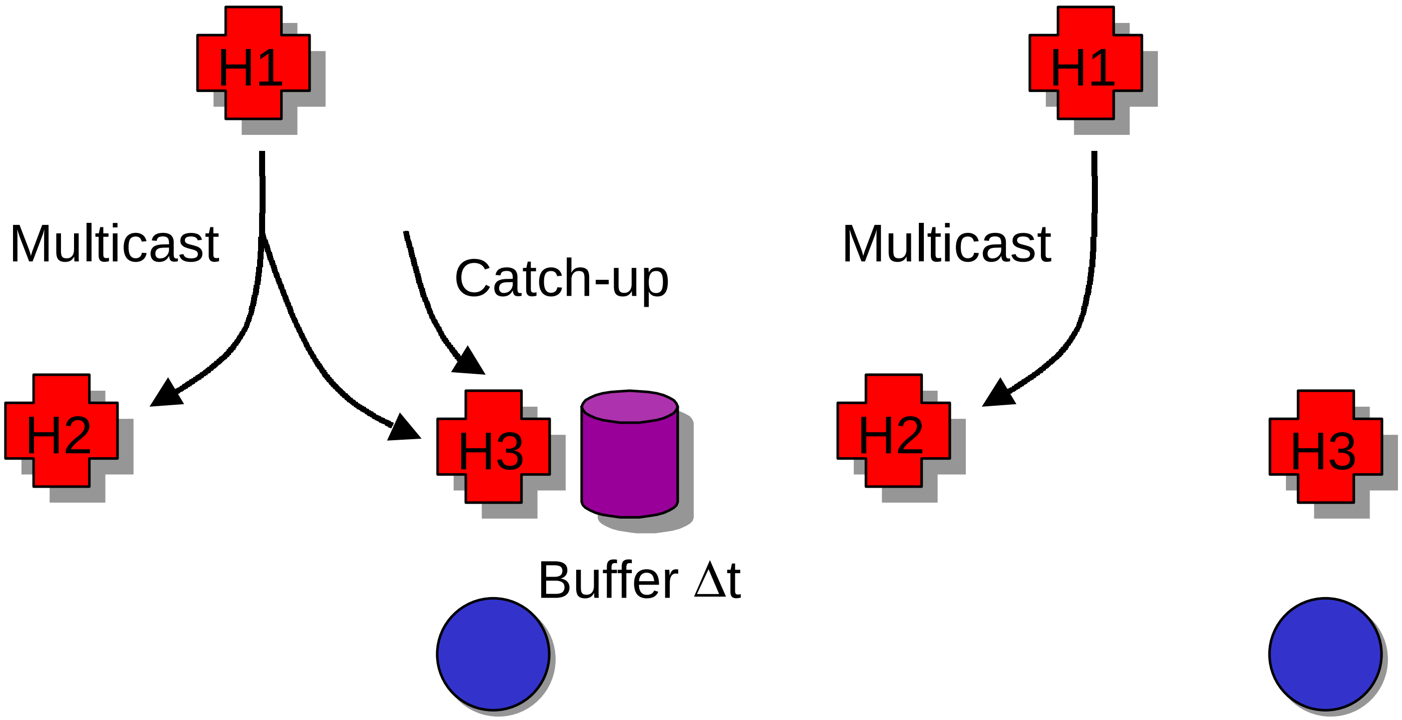
Data Stream Sharing or Not



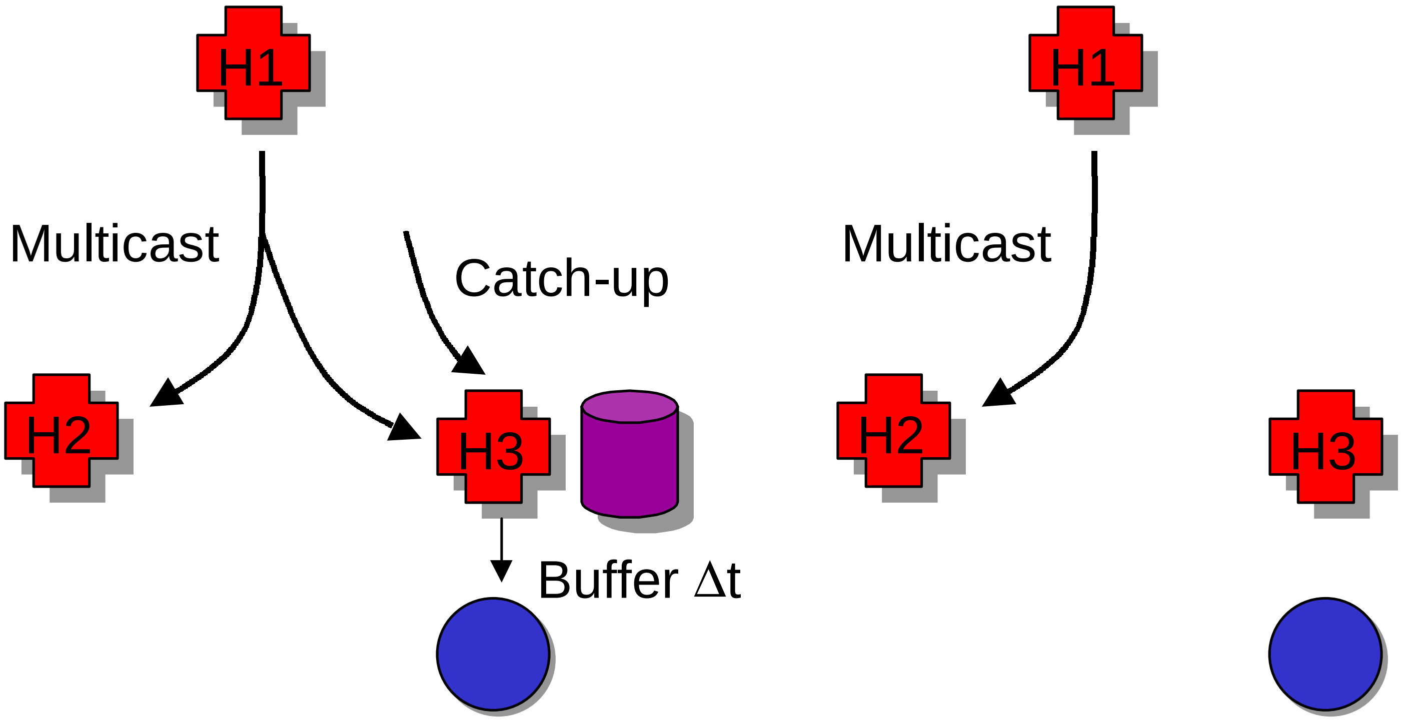
Data Stream Sharing or Not



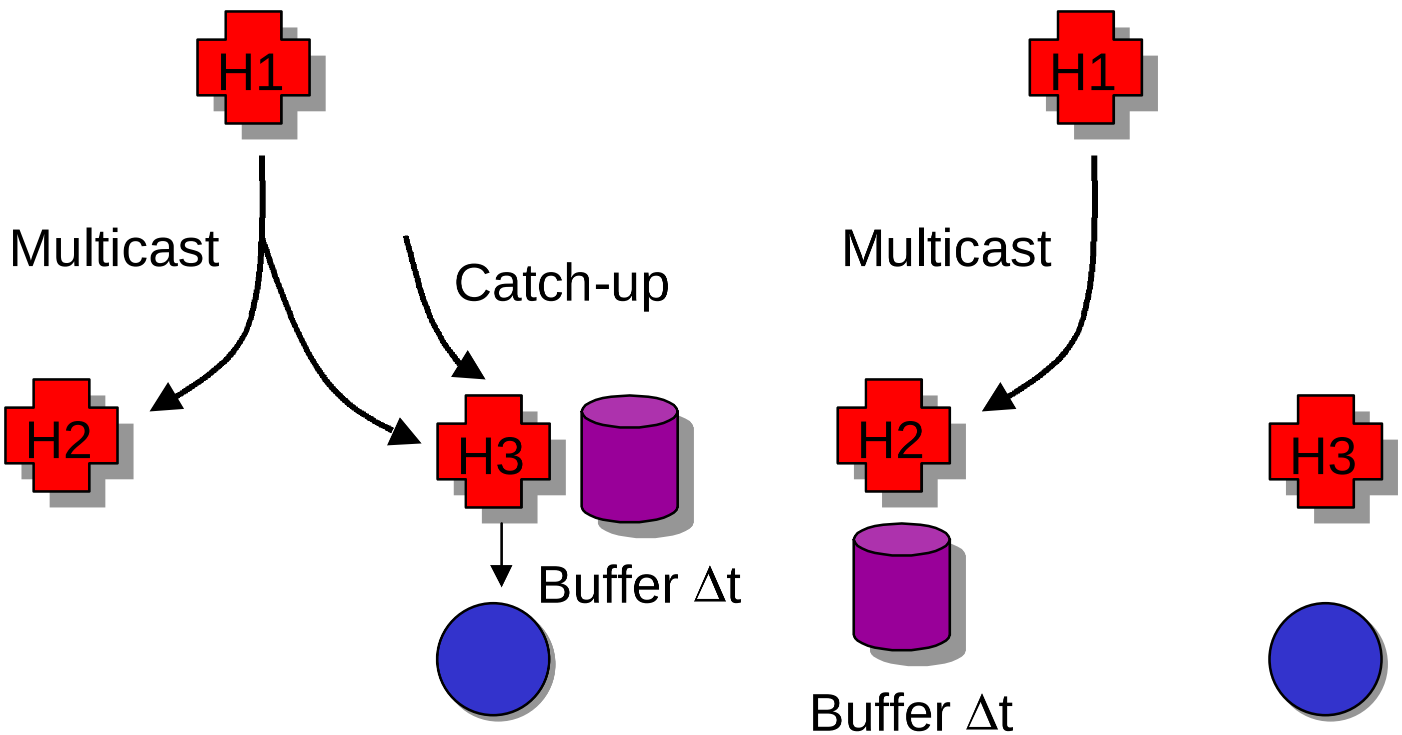
Data Stream Sharing or Not



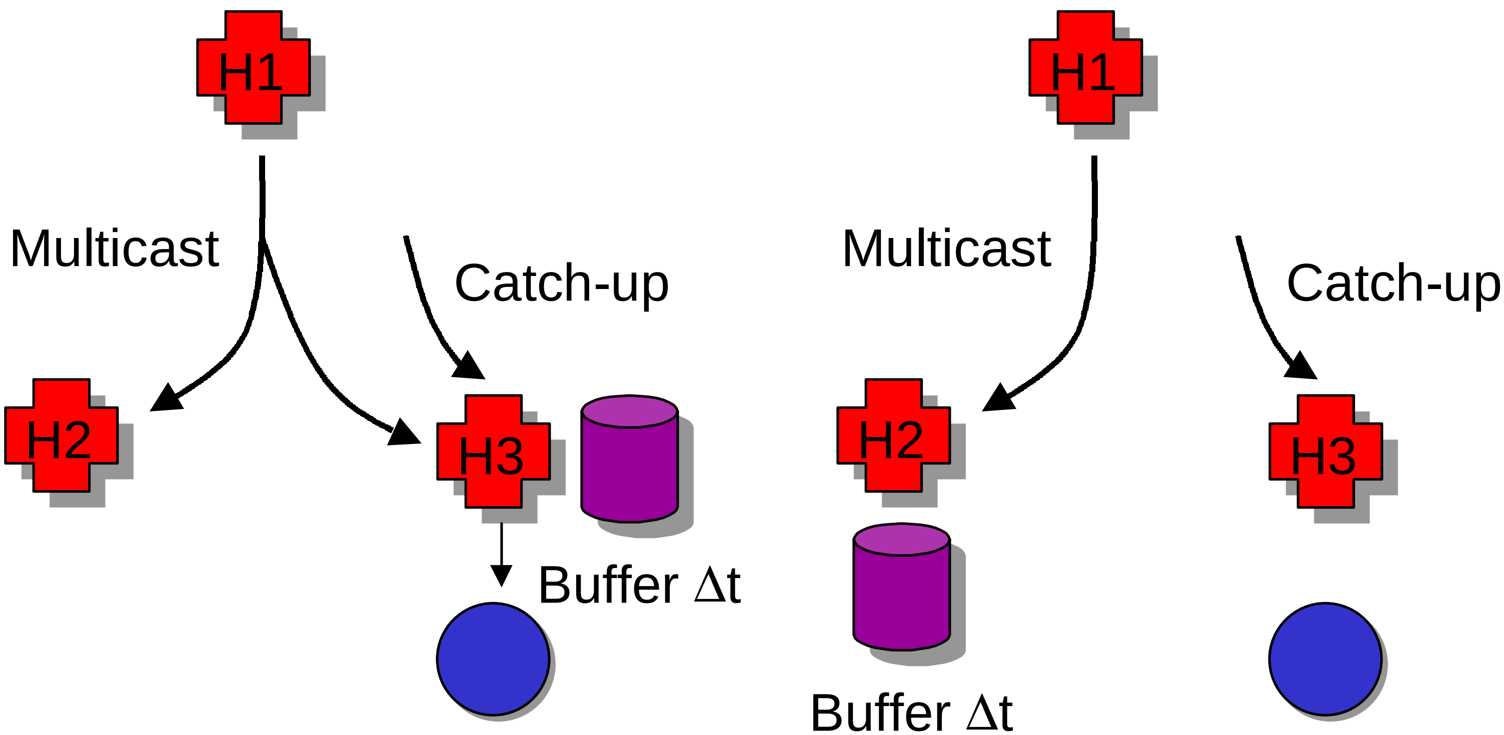
Data Stream Sharing or Not



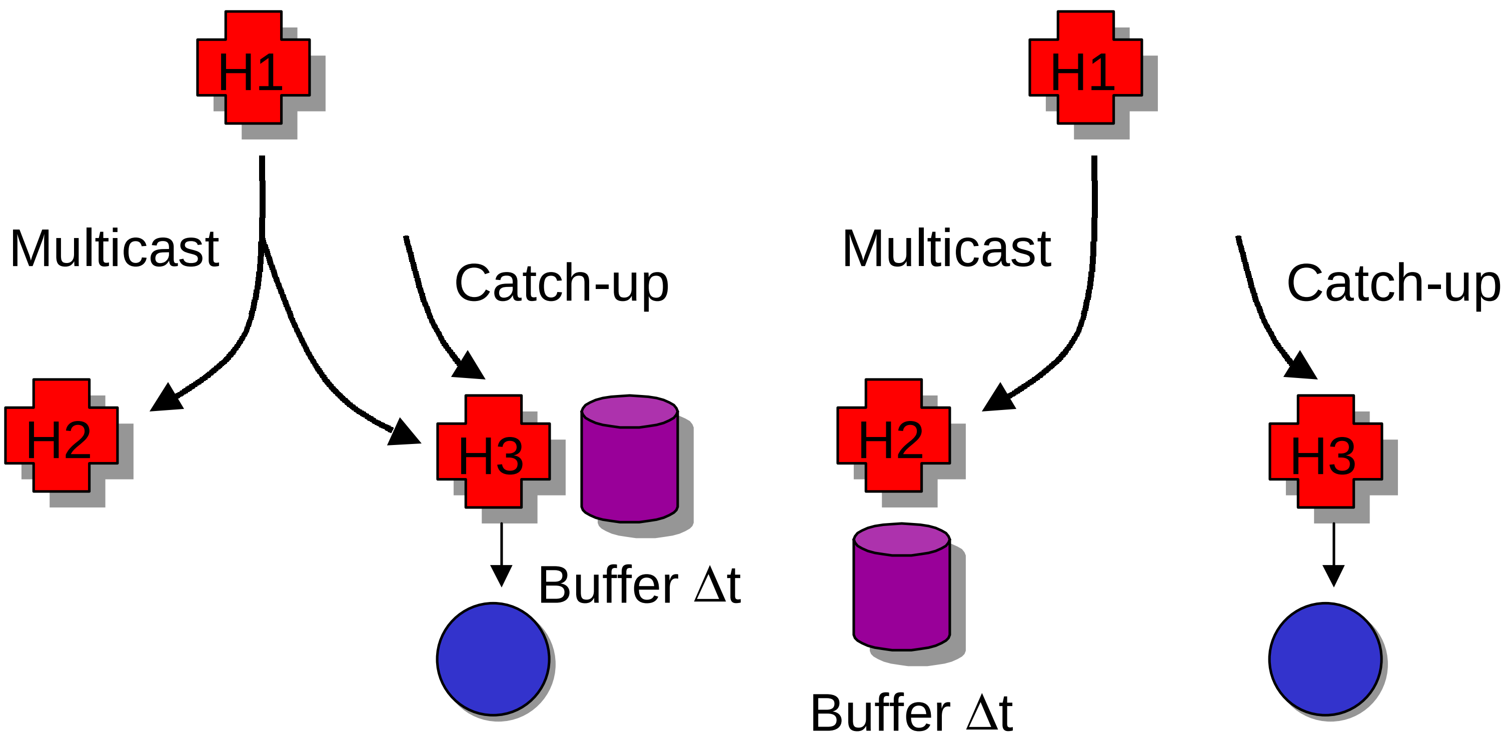
Data Stream Sharing or Not



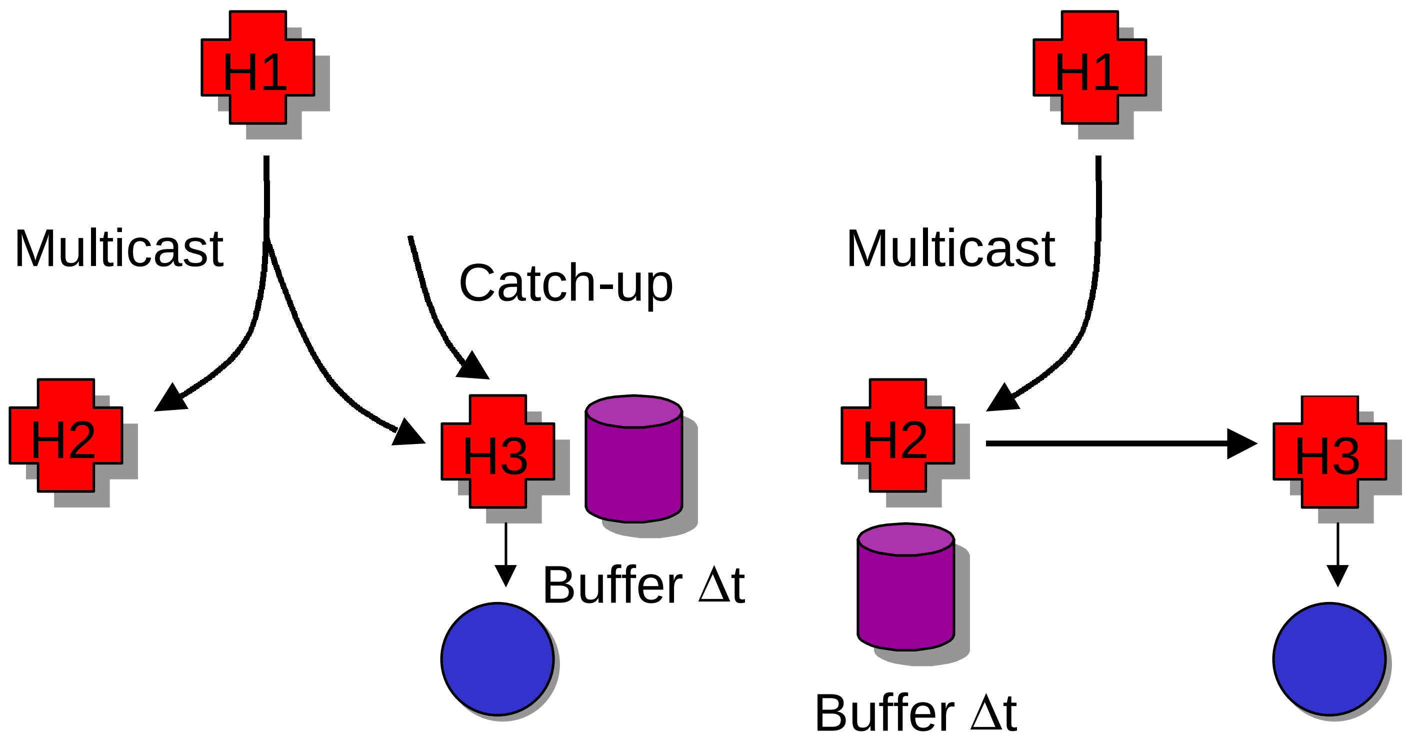
Data Stream Sharing or Not



Data Stream Sharing or Not



Data Stream Sharing or Not

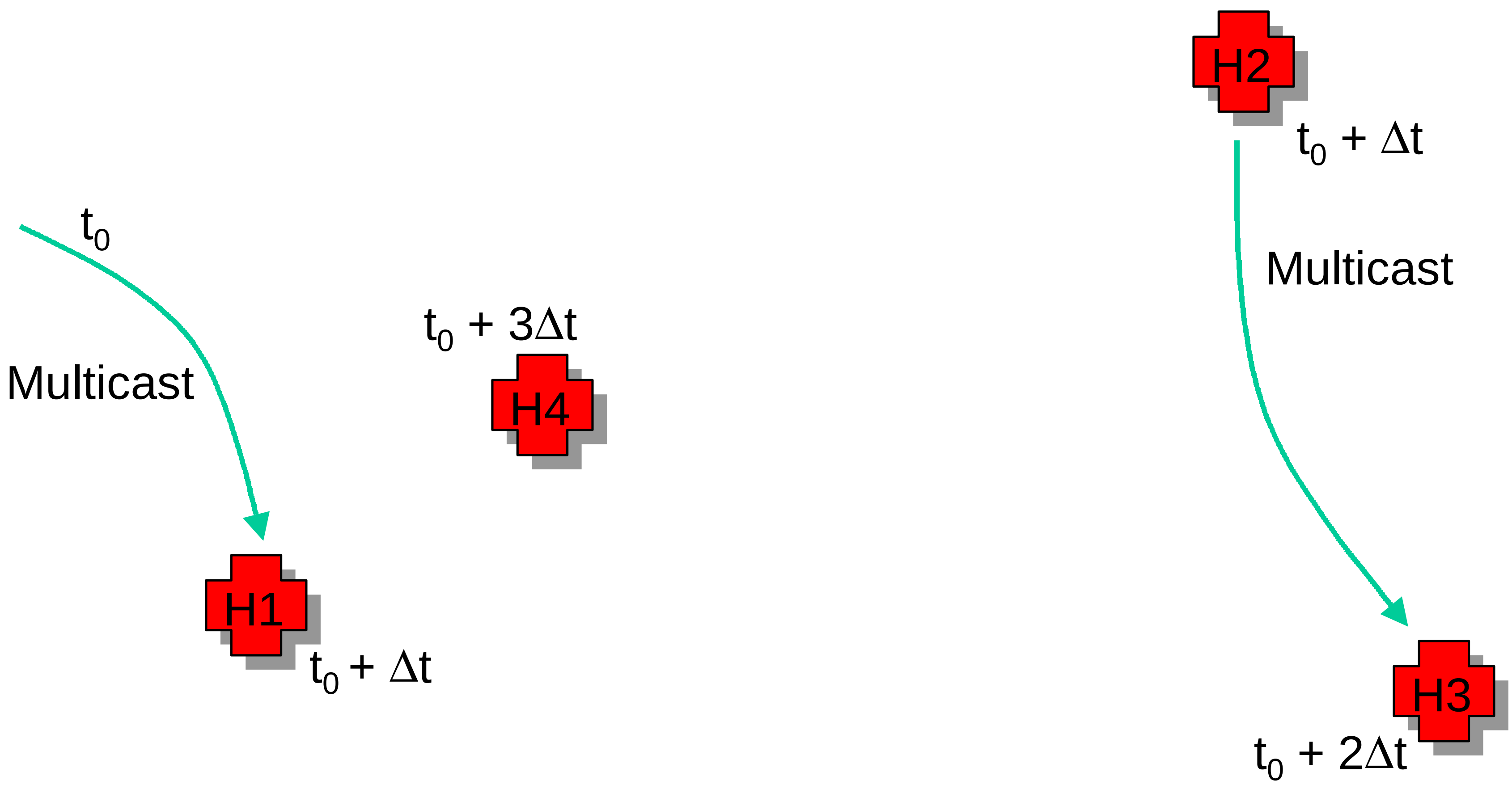


Example Helper Selection Algorithm

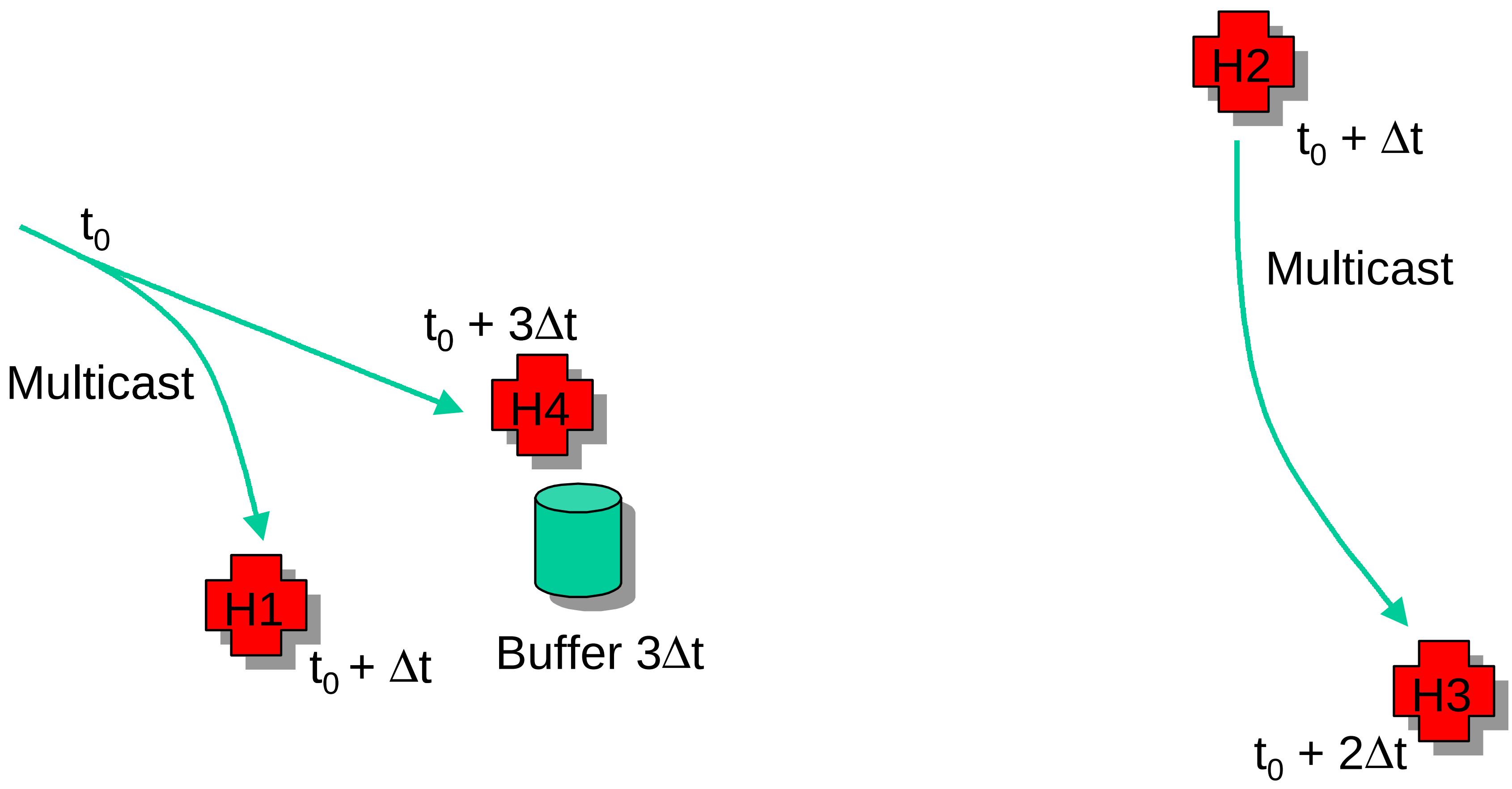
```
foreach region (local, regional, national, global) {
        if (! sHelper) {
                sHelper = findBestStreamSharingHelper(region);
        }
        if (! nHelper) {
                nHelper = findBestNewStreamHelper(region);
        }
        if (sHelper && nHelper) { break; }
}

if ( bufferReq(sHelper) * netDistance(sHelper)  <
   bufferReq(nHelper)  * netDistance(nHelper)) {
        return sHelper;
} else { return nHelper; }
```

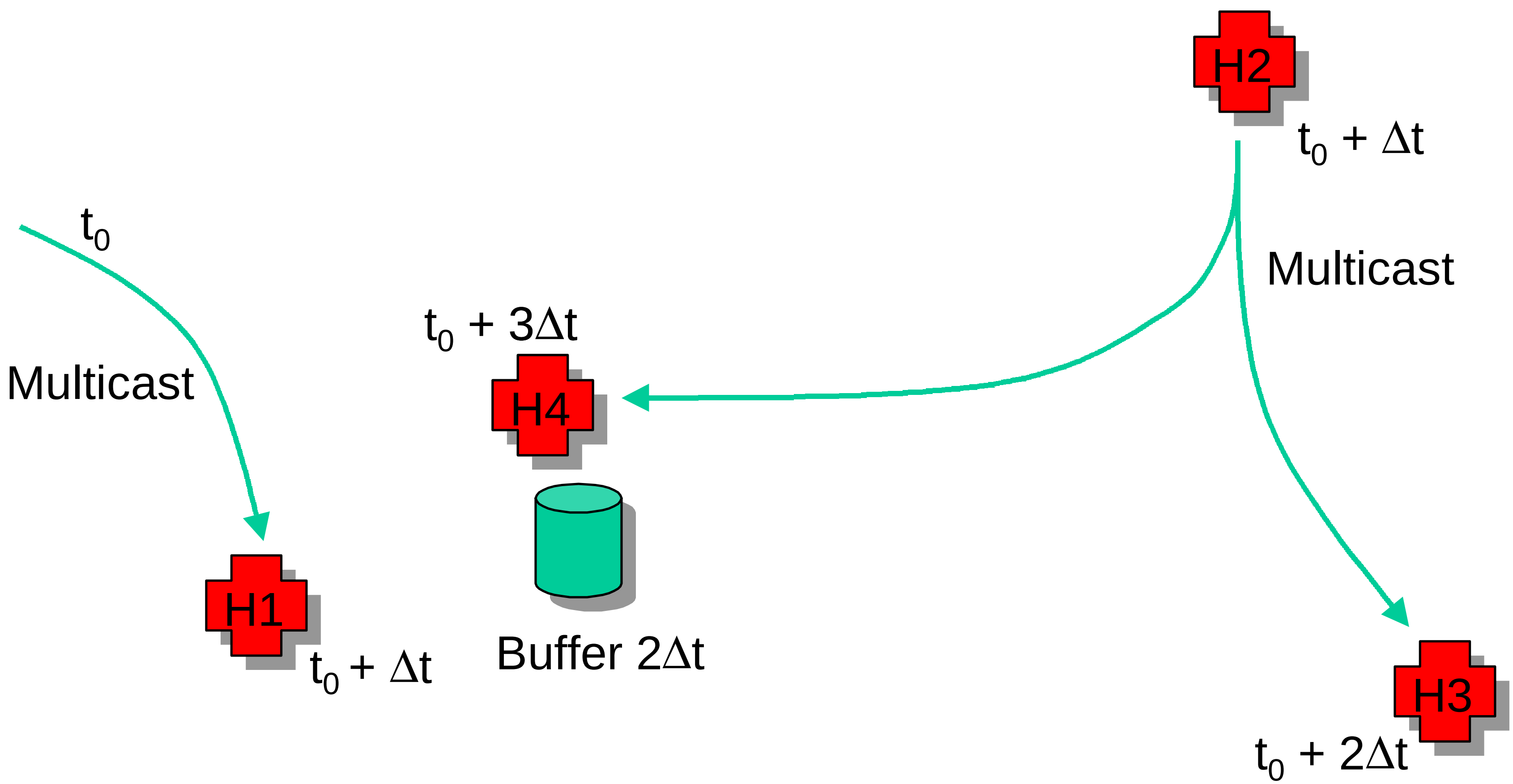
Algorithm in Action



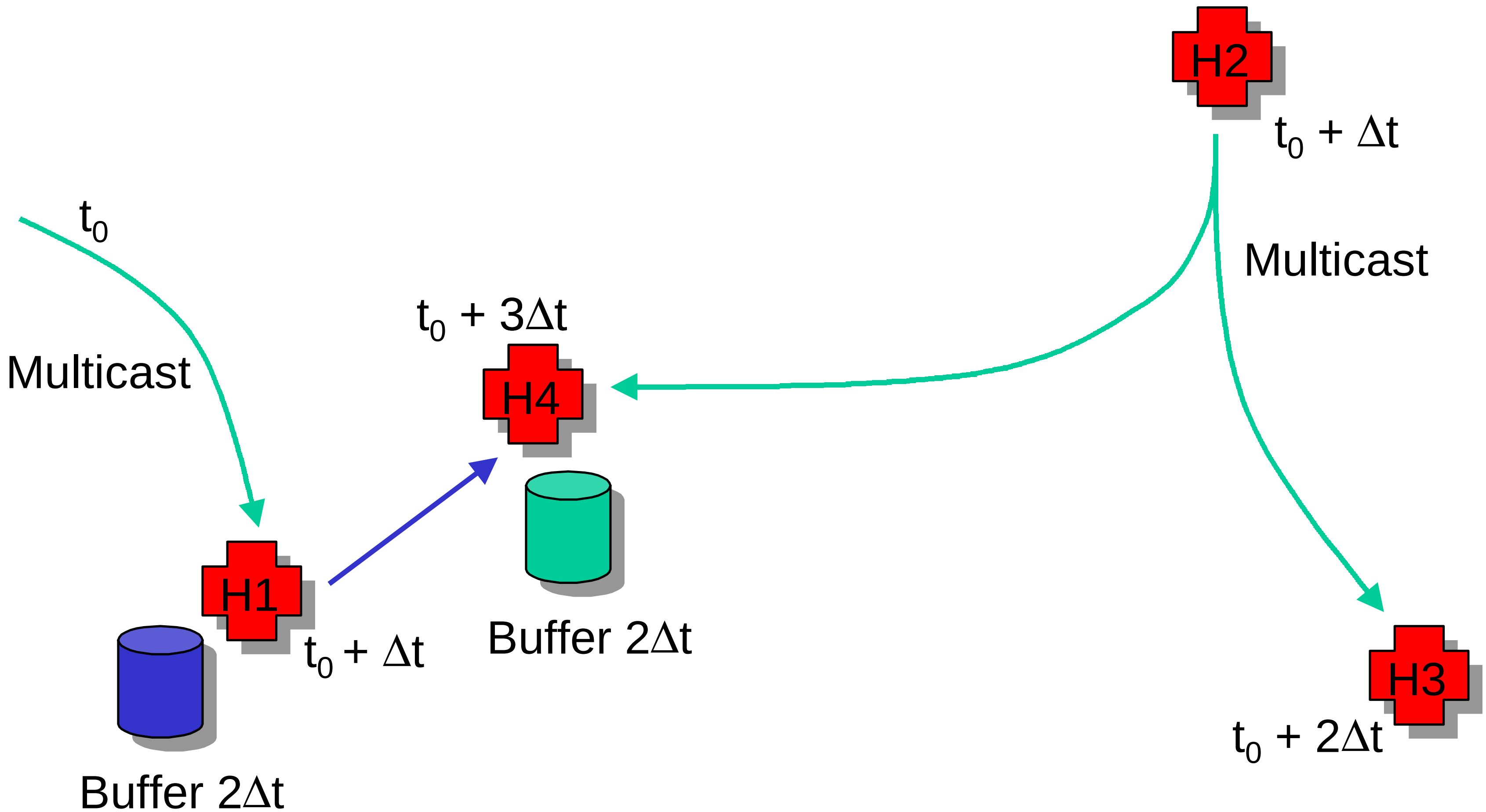
Algorithm in Action



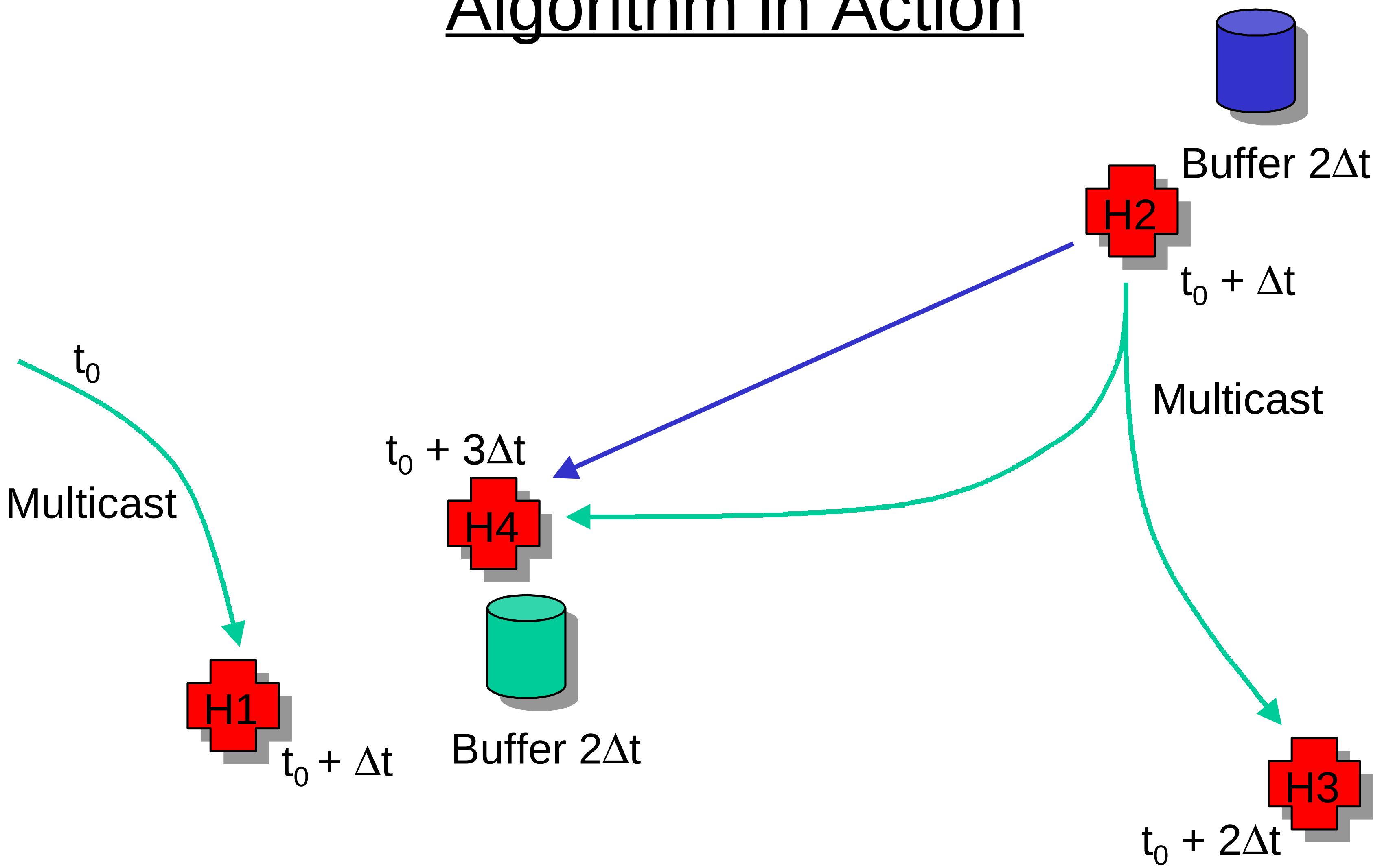
Algorithm in Action



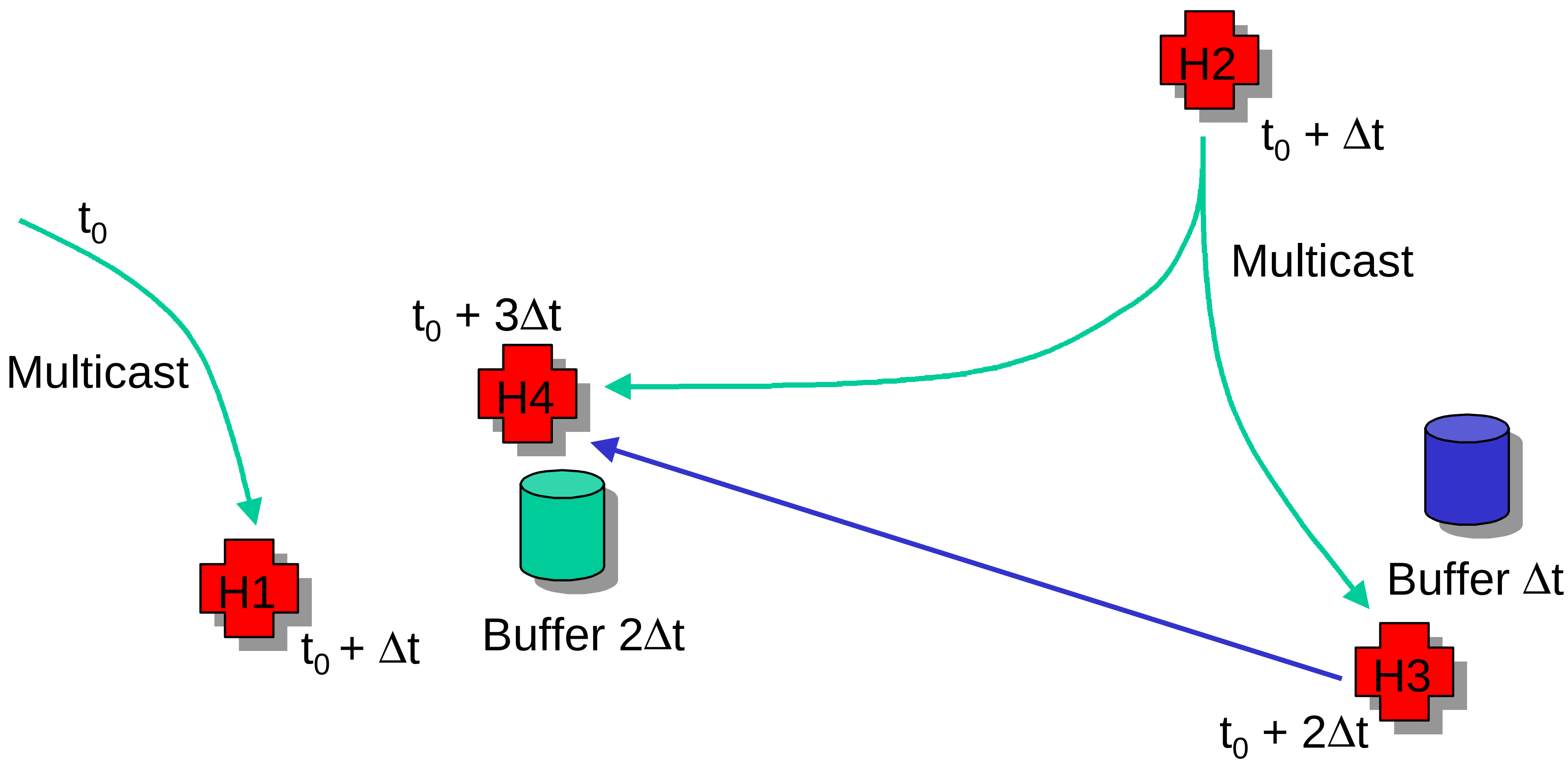
Algorithm in Action



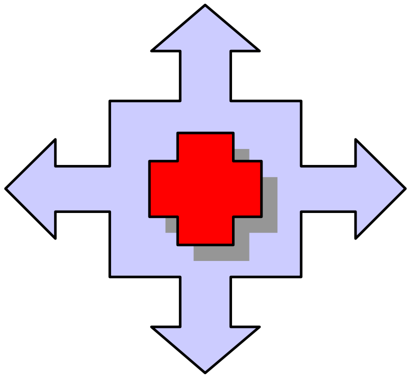
Algorithm in Action



Algorithm in Action

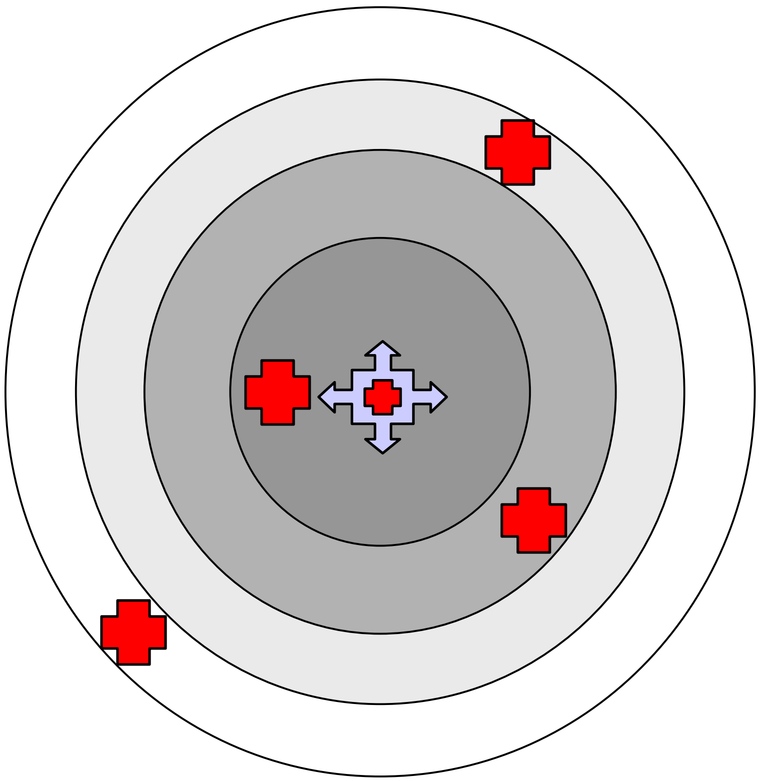


State Distribution: What?



- Streams currently buffered, for each stream:
  - Current lowest segment sequence number
  - Bandwidth of this stream
  - Source of this stream: Multicast vs Unicast
- Overall buffer used and free
- Overall bandwidth used and free

State Distribution: How?



TTL Scope	Frequency of Reception
15	Every Advertisement
31	Every 2nd Advertisement
63	Every 4th Advertisement
127	Every 8th Advertisement

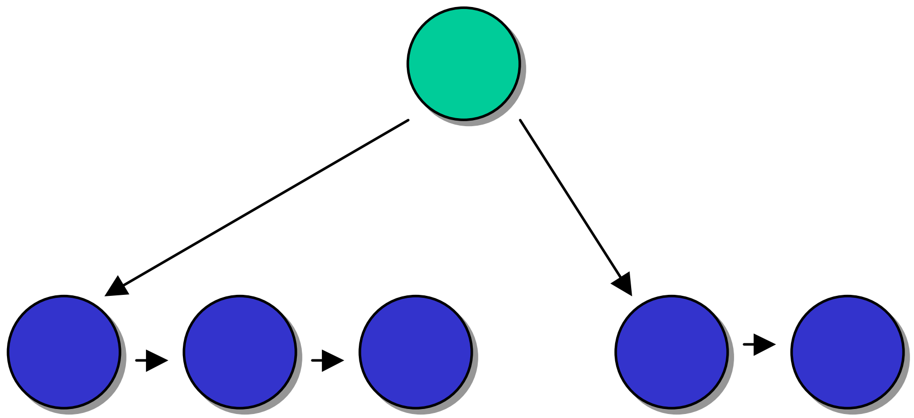
Listening Helper

Advertising Helper

Static Caching of Segments

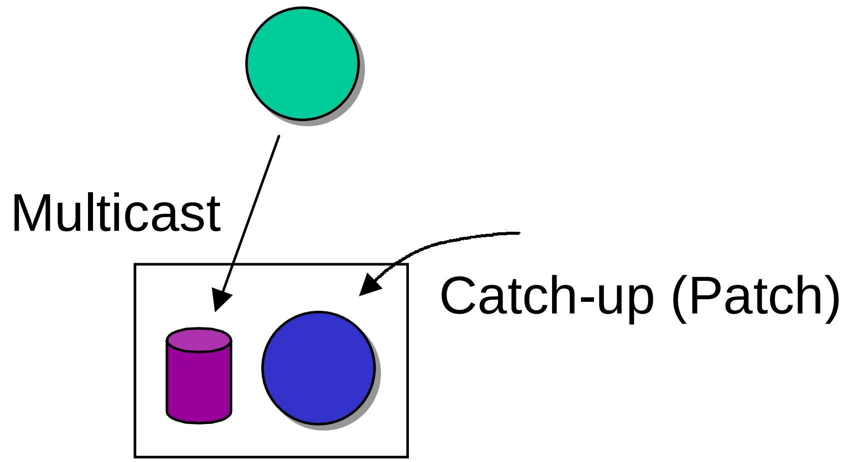
- Short clips
- Hot segments
  - e.g. A particular song in a concert
- Initial segments
  - If receivers tend to start listening from the beginning
  - Further reduce start-up latency
- Catch-up data
- Requires a measure of “hotness”

Related Work



- Chaining [Hua, Sheu and Tavanapong, ICMCS '97]
  - Every receiver buffers a fixed amount of data
  - Late comers can be served out of receiver buffers
  - Forms a chain of receivers
  - Focuses on reducing server load

Related Work



- Patching [Hua, Cai and Sheu, ACM Multimedia ‘98]
  - Each receiver allocates buffer to catch-up with an on-going multicast session originating from the server
  - Patching refers to the need for catch-up data from the server
  - No buffer sharing, data always originates from server

Conclusion

- By using Helpers as data forwarding, buffering, and caching agents, we believe
  - Streaming multimedia quality can be enhanced
  - Server load can be reduced
  - Network load can be reduced
  - It's a win, win, win
- Keys to success
  - Access pattern allows data sharing
  - Low mesh setup overhead, responsiveness is critical
  - Low state distribution overhead
- Huge design space remains to be explored
- System is currently being implemented in ns-2