Exercises 5 & 6: Interaction

Group 1

Editorial Notes

- Added better explanations to the critical reflection in Task 1
- Fixed one choice which we had remembered incorrectly when writing the original document
- Made it prettier :)

Task 1: Critical Reflection of Collective Decisions

Choice	Advantages	Disadvantages
To only notify the winner of the auction	- This decision was based on the fact that we don't intend to keep track of the auctions we bid on because we don't bid any kind of currency. This means that the bidding workflow is very simple to implement for the bidder.	- We are committing to a path that does not support bidding with currency very well since you need to know if you lost an auction and you can free up the currency bid.
To send the whole task to the auction winner.	- Fewer requests and therefore slightly better performance as the task does not have to be fetched in a separate request. - We don't have to give external systems access to look up tasks which means we have a slightly smaller attack surface (we still have to give them patch permission)	- Forces additional coupling between the Auction House and Task List as the Auction House has to fetch the task from the Task List to send it to the auction winner (unless you make the Task List send this to the external auction house which would also be a weird workflow
Tasks are patched by the external group that executed it	- Simple to implement since we are already doing this locally	- Security issues as we have to open this up to external systems. Increases out system's attack surface
Defining JSON schemas for each of the types of messages we send to each other	- Improves interoperability	- Restricts functionality
Changed the auction deadline from seconds represented by an integer to a Timestamp	- It is clear when the auction actually ends as seconds may have passed between the auction being started and the message being received by other auction houses	

Task 4: W3C WebSub vs. MQTT

Technology	Advantages	Disadvantages
WebSub	- Works via HTTP protocol -> similar to interacting over HTTP since you don't need an external protocol - HTTP is used a lot and more common than other protocols which helps with understandability and integration	- Understanding the implementation is hard because it is not widely adopted and there is a lack of online resources, communities, and documentation - No good open-source hub around WebSub. For example, Google's hub is close-sourced - HTTP headers are quite large - You must add handshake implementation manually
MQTT	- Very light weight with small headers - Widely adopted and well documented online - A lot of built-in functionalities like retained messages, persistent clients, and last will & testament offered out of the box - TCP has built-in handshake implementation	- Since you have to communicate through a central broker typically placed in the cloud, latency can become an issue

Comparison - when to use which

- MQTT is preferable when communicating with IoT devices. There are many reasons for this including packet size and QoS guarantees. This could be important in the context of Tapas if we want to implement more IoT executors
- MQTT is also preferable when you want to use the built in features of retained messages and persistent clients
- External communication between different (web) systems: you already have HTTP endpoints set up which WebSub could reuse for different purposes
- MQTT has a bigger community and more open source brokers (hubs) and could therefore be beMer for small projects that want to run on community licences

Conclusion

Due to the advanced built-in functionality of MQTT, we want to use it for internal communication, also in order to replace some services already implemented. Moreover, due to MQTT's lightweight IoT-friendly capability (handling lots of lightweight executors), we want to use MQTT for our executor communication. We would want to use WebSub for external communication, but this decision would depend upon what the other groups think to ensure interoperability.