Technische Universität München
TUM School of Computation, Technology and Information

Embodiment in Virtual and Mixed Environments

Sandro Weber

Vollständiger Abdruck der von der TUM School of Computation, Technology and Information der Technischen Universität München zur Erlangung eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitz: Prof. Dr.-Ing. Georg Carle

Prüfer*innen der Dissertation:

1. Prof. Gudrun J. Klinker, Ph.D.
2. Prof. Dr. Dieter Schmalstieg

Die Dissertation wurde am 09.12.2022 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 09.02.2023 angenommen.

Abstract

The prospect of reality being complemented by a virtual sphere where humans are naturally embedded in their digital surroundings has inspired human imagination in many ways. It carries the promise of transforming all aspects of human life from health care to education, industry and entertainment on a personal to a global level by making them more efficient, less cumbersome or broadening and enriching their scope, influences and possibilities via far-reaching virtual means. The advancements in the fields of Virtual, Augmented & Mixed Reality, Ubiquitous Computing, Internet of Things, Machine Learning among others bring the realization of such scenarios ever closer.
Yet such transformative processes also carry a lot of risks and considerations. Apart from security and privacy concerns similar to today's problems, the process of mixing different environments and interfering with aspects of human life can have non-trivial social, psychological and cultural consequences.

This work presents a framework designed for Ubiquitous Mixed Environments. The framework is built to be generally applicable on all scales, yet it understands itself as one piece to the puzzle: enabling flexible communication between interaction devices and facilitating individualistic setups that allow users to control where and to what extent interaction happens by forming a well-attuned and personalized digital "skin" or "suit" that embodies them in a virtual environment.
The thesis gives a technical and conceptual overview of the framework and argues design decisions. Tests and tools aimed at assisting use and development are also presented.
It then explores use-cases with a focus on embodiment scenarios and how to apply and support modular and interoperable solutions as part of an open Mixed Reality world.

Zusammenfassung

Die Aussicht, dass die Realität durch eine virtuelle Sphäre ergänzt wird in der Menschen auf natürliche Weise in ihr digitales Umfeld eingebettet sind hat die menschliche Fantasie auf viele Weisen inspiriert. Sie bringt das Versprechen mit sich alle Aspekte des menschlichen Lebens - von Gesundheitsvorsorge über Bildung bis hin zu Industrie und Unterhaltung - auf persönlicher bis globaler Ebene zu verändern indem sie effizienter oder weniger beschwerlich gestaltet werden und ihr Horizont, die Einflüsse und Möglichkeiten durch weitreichende virtuelle Hilfsmittel erweitert und bereichert werden.
Jedoch tragen solch umgestaltende Prozesse auch viele Risiken und Bedenken mit sich. Neben den Anliegen zu Sicherheit und Privatsphäre die sich ähnlich zu heutigen Problemen äußern kann das Vermischen von unterschiedlichen Umgebungen und der Eingriff in Aspekte des menschlichen Lebens nicht-triviale soziale, psychologische und kulturelle Konsequenzen haben.

Diese Arbeit präsentiert ein Framework, das für Ubiquitous Mixed Environments entworfen wurde. Das Framework ist für generelle Fälle auf allen Ebenen anwendbar, es versteht sich jedoch als ein Teil des Puzzles: das Ermöglichen von flexibler Kommunikation zwischen Interaktionsgeräten und die Förderung individueller Konfigurationen die dem Nutzer Kontrolle darüber erlauben wann und in welchem Ausmaß Interaktion geschieht indem sie eine gut abgestimmte und personalisierte digitale "Haut" oder digitalen "Anzug" erzeugen, die/der den Nutzer in einer virtuellen Umgebung verkörpert.
Die Arbeit gibt einen technischen und konzeptuellen Überblick über das Framework und verargumentiert Entscheidungen zum Design. Tests und Werkzeuge mit dem Ziel die Nutzung und das Entwickeln zu erleichtern werden ebenfalls präsentiert.
Anschließend werden Anwendungsfälle untersucht mit dem Fokus auf Embodiment-Szenarien und wie modulare und kompatible Lösungen im Rahmen einer offenen Mixed-Reality-Welt angewendet und unterstützt werden können.

Contents

Acknowledgments

Firstly, I want to thank Prof. Gudrun Klinker for her persistent support, for her investment of time and her exceptional care during this endeavor.

Thanks also goes to my mentor Dr. Patrick Maier for being one of the most energetic and enjoyable people to work with.

A lot of appreciation also goes to my colleagues, who are always up for endless discussions and feedback and who took the time to give half-baked ideas a try.

I want to thank family and friends for their continued support and understanding, listening to ideas and offering their viewpoints.

1 Introduction

1.1 Original Motivation

This endeavor originally started with the Neurorobotics Platform (NRP) as a simulation platform for embodied robotics and the goal of letting humans join the virtual robots in their simulated environment through the use of commercially affordable and available hardware.

The focus was to establish both robot and human as equal actors on a level playing field inside the simulation without one having privileged access to parts of the physics or visualization nor being able to circumvent any of their internal mechanisms - for example observing elements or performing actions that would be considered physically impossible.
For a human user to be able to present themselves with agency in this manner inside the virtual world of the robot, they would require a surrogate body - an avatar - embedded in the physics engines of the simulator that they can take over and control.

Given such a setup, it would allow for investigations in two directions. Firstly, as the robot is being developed and tested inside this simulated scenery during the first phases of its life-cycle, it would be able to encounter human behaviour and interaction with more of its complex and unpredictable facets - something inherently difficult to simulate. Secondly, it would explore what is necessary and what it means to bring agency to humans in a virtual environment when they are not the center piece but instead just one more actor inside it - being exposed to all the limiting factors like physics calculations affecting their proxy body etc. that keep the virtual world consistent while trying to approximate reality.

Due to several technical as well as political decisions during the development of the NRP it became clear that real-time and interactive simulations were not the primary consideration for the foreseeable future, i.e. the time-frame of this work. Non-real-time simulation however is an obvious death sentence to proper experiments involving human interaction in Virtual Reality (VR). In consequence, the scope of this work changed to finding an approach that would allow to

- quickly set up an alternative environment involving the same base components like physics engine, VR rendering, robot sensors & actuators, etc.
- being able to adopt the already invested work for the NRP in terms of VR rendering, human avatar motion controls, etc. as much as possible
- allowing this separate solution to hook back into the original platform at a later stage if possible

- work in conjunction with all state-of-the-art utilities for VR and robotics development and hopefully future developments which are occurring at a fast pace

Following these considerations, it became clear that fulfilling all these wishes would require a framework that was highly configurable and modular. The impression also formed that its use might well extend beyond the scope of this work alone and the base necessary mechanisms could also prove helpful in other applications in the field of Mixed Reality (MR) in general.

1.2 Generalizing the Original Goal

All of these considerations shifted the goal of this work towards a more general approach of a framework designed for MR scenarios. The focus was on building connections and interactivity between heterogeneous and distributed systems, trying to find solutions with the highly dynamic situations that are inherent to MR.

To elaborate, let us extend the idea of bringing humans and virtual robots together into the same room by reversing the roles of human and robot: we try to bring the simulated robot into the real laboratory by means of augmented reality.
Robots already are highly complex systems with physical and virtual parts to them, so they seem like a good starting point for thinking about how these two worlds can mix. We could opt to build a sort of proxy platform that acts as a real-life representation for the robot so it can actually physically influence its environment. The closer that proxy gets to the actual capabilities of the robot the more we are actually just building the robot in reality. This line of thinking relates to the concepts of a digital twin, only it is approaching from the opposite side and instead tries to establish a "real-life twin".
If we instead keep the robot purely virtual we would nevertheless need some sort of room-scanning device creating a virtual representation of the environment that we then can (partly) convey to the virtual robot as its point of view.

In the reverse case we seem to have a much easier time supplying a virtual body to a human as the virtual world usually lends itself to be our playground with little limitations. Trying to truly encompass the human capabilities as was argued for the robot proxy before however is just as daunting of a task on closer inspection.
To map all desirable forms of interaction and cover its full range, it would require technology almost forming a "digital skin" for an individual user - involving smart software that can translate and mitigate between human body and virtual representation and environment, operating in a tight loop with the user and acting as a sort of digital cognitive extension managing parts of the virtual

Figure 1: "Hyper-Reality" by Keiichi Matsuda.

aspects.
Such a system does not have to be limited to virtual extensions/augmentations either, it could be similarly applied to real-life robotics (see 3.1.5).

Efforts like Nvidia's *Omniverse*[1] or Meta's *Metaverse*[2] indicate that this merging of realities is of great interest to global companies for various purposes and forms of activity.

The concept film "Hyper-Reality"[3] by Keiichi Matsuda poignantly paints a picture of a ubiquitous case where realities overload each other (Figure 1).
In such an environment, how is it decided what is important, worth attention and visualization and/or put into the foreground? How is it decided what to interact with and how?

[1] `https://www.nvidia.com/gtc/keynote/` (last accessed 08-06-2022)
[2] `https://about.facebook.com/what-is-the-metaverse/` (last accessed 08-06-2022)
[3] `http://km.cx/projects/hyper-reality`

2 Theoretical Background: Mixed Reality

Coming from the introductory scenario, the goal is to build a framework for Ubiquitous Mixed Reality (UMR) and explore degrees of virtual embodiment. This section will provide an overview over the terms Virtual Reality, Mixed Reality and ubiquitous computing to see how they are defined, what different interpretations are offered, what they imply and what conceptual boundaries can be identified to form a better understanding of where the eventual framework is placed and what it is trying to achieve.

2.1 Motivational Scenario Revisited

As a thought experiment, let us examine the virtual embodiment scenario of a human in a virtual simulation from the introduction again and extend it with additional hypothetical features to achieve a closer investigation of our terminology.

In its original conception, it is as close to the colloquial interpretation of VR as today's consumer hardware will allow us to go: a human joining a virtual and fully simulated environment where things continue to happen even without their involvement.
Once the human has a virtual body - or at least something that can replicate the functionality of a body part - that is adhering to and embedded in the virtual laws (of physics), they can use it to "get in touch" with the virtual world.

The robot on the other hand perceives its environment through its simulated sensors and acts through its simulated actuators. The processing of its sensors/actuators (cognition may be a bit of an overstatement), too, is entirely reliant on synthetic computation via a brain simulation or simpler algorithms. What do we call the reality the robotic system is processing? If we regard the robot as its own individual system - as unintelligent as it may be - that has to find its way around an unknown environment by processing signals and we locate it on the virtual end of the spectrum, the argument is that it perceives *true* or the purest form of VR: there are no physical components influencing its perception, it is a purely simulated system existing in a purely simulated environment performing simulated processing which forms its subjective simulated (or virtual) reality. Whatever technology a human would utilize, their experience of that same reality would be a more mitigated and intermediate one with remaining other-reality-influences compared to the robot's perception.

What if we then tried to turn the physical-virtual dials for different aspects of this scenario? What if we wanted to replace the simulated environment with a real laboratory or robot test course? We could generate 3D scans of the physical environment or try to generate live 3D mappings of the surrounding geometry

via SLAM algorithms ([1][2]), then place the virtual robot inside it. Instead of a virtual robot we could alternatively imagine an intelligent virtual assistant (IVA, see 4.5) in the same room.

Now the robot is facing the same issue as the human in the initial scenario - it does not have a body (sensors / actuators attached to a platform) it could use to interact with its environment.
If the 3D environment mapping is detailed and complete enough, we could emulate a virtual camera perspective reflecting the robot's location and feed it to the robotics internal systems. Checking the virtual robot's geometry against the 3D mapping, we could also tell it when it supposedly bumped into a wall or collided with objects. Given the live updates of the environment geometry happen fast enough, this might even be possible for collisions with mobile entities like real humans in the room. IoT devices located in the room could give the robot additional impressions about its environment like temperature which it might otherwise get from its on-board sensors.

But to have real physical presence and to be able to, in turn, physically affect its surrounding the robot would need a physical body or access to e.g. actuators/effectors. If not that, then it would have to rely on a system capable of emulating certain forces - in a simple form this could be achieved via physical proxy objects and more dynamically via robotics systems moving and providing appropriate haptic/force feedback throughout the physical environment. [3][4][5]
For the robot, we could start with a little proxy construction - just a platform with wheels in the approximate dimensions and a stick with a camera attached. We could extend this proxy until we realized the full robot. Now we introduce a second and a third robot, which have not been fully realized and still rely on physical proxy elements or are purely virtual. Maybe we have a person that is physically present in the lab and another human joins the scenario through virtual means using VR technology.
How about we throw a ball that is purely virtual into the scenario for all robots and humans to play with together - the ball of course having virtual material properties (elasticity, surface friction, etc.) and reacting to the physical environment (carpet floor, cement wall, etc.).
As the virtual environment would be replaced with scans of a physical environment or the robot being actualized with e.g. a camera, it would translate towards physical reality (PR) in its experience and be part of a MR too - only that it approaches the Milgram continuum[6] (see Figure 2) from the other end. Can it translate fully to PR or will some minimal part of its perception always remain VR as with humans utilizing a CAVE or HMD setup?

This gives the impression of several environments where all entities involved are heterogeneous and complicated or even complex systems[7] in and of themselves, trying to perceive their environment(s) and interact with it. They, as well as other entities, might exist on different ends of the continuum, maybe they

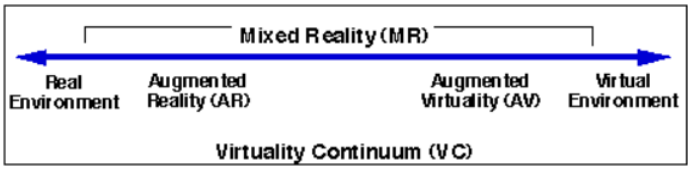


Figure 2: Milgram-Kishino continuum for categorizing AR, VR, MR. [6]

are comprised of physical and virtual components at the same time, maybe the nature of what currently represents them changes over time and depends on the circumstances.

Perception and interaction with the other environment can only happen if mechanisms can be found to translate/mediate input & output and if rules of causality between environments can be established. The intra- as well as inter-environment rules of causality constitute action-reaction relations and define possible interactions.
Some interactions we can choose and define ourselves, some are dictated onto us. We can also see that there are some discrepancies and limitations between environments that are not solvable and make interactions impossible with currently available technology - as in the case of a system applying forces or emulating physical barriers where current technology can only fake/emulate so much until it becomes an intrusive hindrance itself.

The mixed environment itself becomes a complex system too, at the latest when it involves influences from ubiquitous computing and becomes a socio-technological system on scales from individual to global involving personalized computation, smart environments, machine learning and the risks that come with such considerations.

2.2 Ubiquitous Mixed Reality & Environments

In 1994 Milgram and Kishino[6] introduced a virtuality continuum along a 1-dimensional axis, particularly focused on MR visual displays. In Figure 2 we see the continuum with real (i.e. physical) and virtual environment lying on the outer bounds and Mixed Reality (MR) spanning all scenarios where some influence or component of both worlds is present, like Augmented Reality (AR) or Augmented Virtuality (AV).

According to Speicher et al. [8] the Milgram-Kishino continuum is still the most cited source when it comes to defining MR. Speicher et al. also state, however, that there is no singular established definition for MR and that varying and sometimes contradictory interpretations exist between experts.
Arguably, MR is very multi-faceted and multi-dimensional in its nature so a

linear scale is not quite adequate when talking about what MR encompasses currently and possibly in the future. If the goal is to build a framework set in the field of MR or Mixed Environments (ME), it is necessary to find at least useful models reflecting central aspects of it. In order to think about these models, an informative (working) definition is helpful or at least should serve to clarify the approach the framework is taking on the matter.

2.2.1 Virtual Reality & Environments

In Milgram et al.'s work on "A taxonomy of mixed reality visual displays"[6] the real environment and the virtual environment are the outer boundaries of the mixed reality continuum. The term *physical* environment instead of *real* environment would be more distinguishing here as the term *reality* is heavily used for all aspects of the spectrum and the interpretation of what is *real* is rather vague.

To expand on the scope of what a virtual environment implies: what is today available as VR applications or hardware is hardly the conceptual boundary of what we can envision as an equivalent reality on the other end of the spectrum. It often reflects the current technical limitations for human-centric immersive 3D technology trying to mimic aspects of our physical environment or substitute it.
But it often also entails a limiting scope when thinking about what a VR/VE means in full consequence. Trying to explore these conceptual boundaries, the common factors and differences between physical and virtual environments and what constitutes an environment in general might give some better ideas for a consistent definition. This work takes the word "virtual"[4][5][6][7] as to mean "artificial", "synthetic", "computer generated" or "only indirectly evident" instead of "decidedly not real" or "fictional". As an additional note, a VR/VE is seen as possibly including very abstract processes not necessarily aligning with familiar physical reality concepts.

There may also arise the notion that virtual stands for whatever is not the original subjective reality, i.e. what is virtual reality from our perspective, for a virtual existence from that virtual reality it is normal reality and them experiencing our physical environment would be their virtual reality. This notion is of course problematic because it confuses terms or makes them difficult to use for consistent communication. In that case one might just label our physical environment as *PE* and the virtual one as *VE* to clarify and ensure the statements made are referring to the same environment.

[4] `https://www.merriam-webster.com/dictionary/virtual` (last (last accessed 08-06-2022)
[5] `https://www.thefreedictionary.com/virtual` (last (last accessed 08-06-2022)
[6] `https://dictionary.cambridge.org/us/dictionary/english/virtual` (last (last accessed 08-06-2022)
[7] `https://www.dictionary.com/browse/virtual` (last (last accessed 08-06-2022)

Due to this vagueness, this work seeks a clear separation of the terms *reality* and *environment*. For example, we have to acknowledge that our interpretation of our physical surroundings as our physical reality heavily relies on our perception through the senses and cognition available to us, mediating signals and forming our interpretation.
Nevertheless, we would certainly not deny a fully blind person to be living in the same objective reality as a person with intact vision as it is possible for both to share and agree upon observations about it, e.g. the pinch of a needle. [9]

The discussion will consequently use *reality* as an indication of a subjectively formed representation using senses and agency within an objective *environment*. One could instead prefix terms with subjective/objective (e.g. sMR / oMR), but *reality*/*environment* are felt to form a clearer distinction and cleaner reading experience.

A very defining characteristic of our physical environment is that we are very much trapped inside it and nothing can transcend its causal effects (to our knowledge). It is the boundary condition for our existence. Another aspect is a certain persistence of entities and actions having considerable consequences. As the ultimate indicator: if the consequences of a process can be so drastic as to cause our death, then we can not help but consider it very much real (also see 3.1).
A more logical equivalent as an analog to our physical environment would be to have a virtual environment with (more or less) persistent entities, properties, events and causality rules, possibly even observers and simulated consciousness living inside it. This environment would not necessarily have to adhere to the same laws and stipulations as our physical environment nor would it have to work on the same time-scale. Consciousness, too, could be very non-human.
To put it at an equivalent scale: if it was possible to simulate a whole planet, star system, galaxy or even universe together with a population of conscious observers that had no need or possibility to escape it, then that simulation would constitute their reality. This comes closer to what one could interpret as a strict or "pure" virtual environment.
Any process inside the simulation may be completely oblivious to what is going on in our external physical environment - the only thing of consequence to internals of the simulation is the outside physical hardware being kept intact and operating. There would be no need or means to escape the simulation.
This existence, in turn, reflects propositions by the simulation hypothesis[10] about the nature and composition of our physical reality - only if we built and ran the simulation, we could be sure about its true nature as a sort of artificial "sub-reality".

Some works like the one of Robert J. Bradbury on Matrioshka-Brains [11] even try to think about how hardware to run simulations on such a massive scale could be theoretically achieved within the constraints of our reality. And while such technologies can assuredly be put off into the far future (if ever

achievable), imaginations and thought experiments along these lines might serve to shift the mindset a little into conceptualizing a world where the virtual is acknowledged as potentially its own form of environment/reality and is not just a single playground extension to hop into and forget about immediately after closing it. One could imagine virtual spheres of existence that persist and evolve with little outside intervention or explicit consent. It may help to recognize the eventual complexity of a space where physical and virtual entities (co-)exist, collaborate, combine, compete and truly mix with each other.

This is not to mean that we should start building VEs for artificial minds completely bypassing humanity. If humans build VEs and use VR, it should be of assistance to humans.
Naturally it will reflect concepts that are familiar and easy to grasp and handle for us like 3-dimensionality, speaking to our human senses and having agency by controlling elements exclusive to us, i.e. having a body as a manifest corporeal representation that grounds us in said environment (3.1).
But whenever an application is primarily geared towards human senses and is not so much interested in establishing something independent that resembles an environment it might be better to use terms like "immersive technology" [12][13] to avoid this interpretation conflict of "reality" vs. "environment".
The mediating step of making certain properties visible, understandable and interactive to humans does not encompass all aspects of what a virtual environment might be, yet is often labeled as VE. In fact, a virtual environment that continues to evolve without a human overlooking or controlling it seems to conform much more with the understanding of "reality" than a user-centric immersive application.

Expert opinions and interpretations

When Speicher et al.[8] interviewed experts to characterize virtual reality, they pointed towards an artificially constructed reality and to fully immersive technology with a focus on head-mounted displays (HMDs) and the ability to visit remote places.

Example: 360° Video
As examples for VR, two experts named watching 360° video using an HMD - whereas watching it on a smartphone 2D display was not considered VR.

Discussion:
This example hints at the important distinction, when there is VR hardware and technology at work but the content is in the form of static playback. One may label it with the term reality as it plays with one's perceived surroundings. Arguably though, excluding outside visual (and maybe auditive) perception of physical reality, and replacing it with virtual content that adjusts analogously with head/posture movements to how physical reality would, does alter a big part of human perception but is not nearly enough to "just ignore" and replace

physical reality altogether. Alternatively one may just call it a more elaborate and immersive presentation of a video playback.

In terms of environment however there is almost no dynamics, it is a static video where entities do not really interact with each other and perform rule/event-based exchanges. The only available influence is being able to adjust one's perspective.
On this note, agent-based modeling and simulations[14] have more landmarks of a virtual environment - even when not displayed in an immersive way - than video playback. This field of research also relies on the term "environment" to describe the space or boundary condition an agent is perceiving and acting within. Examples given in [14] of environments for agents are the physical world, a user via a graphical user interface, a collection of other agents, the internet, or a combination of them. Agents themselves are characterized with the properties of autonomy (internal state and decision making without direct intervention), reactivity (situated in environment, respond in a timely fashion), pro-activeness (capable of initiative, not just reactions) and social ability (interaction with other agents and humans through an agent-communication language, achieve goals with the help of cooperation or negotiation).
These characteristics for agents also serve well to classify how dynamic a VE is. The more agent-like elements a VE is comprised of, the more the VE will turn from a static background into a system with momentum that is capable of producing more wide-ranging or variable consequences when interacted with. In consequence, mixing with such a VE produces more involved results and possibilities.

Example: CAVE
Another example for VR named was a CAVE system[15][16].

Discussion:
CAVEs - just like HMD-based systems - serve to illustrate the aforementioned point: they may immerse the user and produce some aspects of VR, but they will not be able to exclude the physical environment nor the implications of the user's body (3.1.1). One can very much run against the walls of a CAVE and hurt one's nose and feet. With HMDs one can still get entangled in video cables or hit objects in the room (or the room itself) if not careful. All these things serve as a very quick reminder about the other environment (and in consequence its reality) we're still beholden to.
CAVE and HMD setups certainly try to minimize any influences coming from physical interactions. But save for a direct VR brain interface with signal exchange blocking and bypassing our normal perception, these influences will always remain to some extent. Even wearing VR equipment and the tiring effects caused by it can break immersion and pull us back to (physical) reality. Later arguments about embodiment (3.1) will pick up on details of this consideration.

Example: Movie "The Matrix"

On the topic of brain interfaces, the movie "The Matrix"[8] was also named as an example of VR. This work of fiction presents a world where humans are held inside a simulated world indistinguishable from physical reality through a brain link. In this fictional work, still the virtual presence vanishes with the death of the physical body.

Discussion:
Here it is important to consider what is being identified as VR - the construct of the Matrix as a shared virtual environment or the brain interface used to enter it. In the first case (VR = VE), this question follows: the virtual reality(environment) in the movie is also occupied by purely artificial minds and agents. If one was to remove all human existences from the Matrix itself and leave its simulation to the machines only, would it stop being a virtual reality? In other words: is the involvement of humans a necessary criterion for a virtual reality and any reality in general?

If VEs are to be interpreted as an outer boundary for a spectrum, the definition of a VE as a "side-reality" that is exclusively dependent on human involvement is not a definition this work follows, as it removes many interesting considerations and possibilities for the modeling of mixed environments (or reality for that matter) - autonomous agents or assistants that remain active when the user is not, among others.
Instead, it is considered as a system that may indeed be very limited in its complexity and time-scale of existence or heavily rely on human input. But any human involvement short of digitalizing a human mind and uploading it should be considered MR with a heavy focus on the VR side - in some cases referred to as AV. Respectively the actual outer boundary lies further off to the side and includes more than immersion of human senses - something that may be implied by the continued arrows in the original continuum ([6]) but not so much in other discussions about the characteristics of VR.

If one would prefer the term *simulated reality* instead of *virtual reality* for this definition, one would have to specify that *simulation* is not meant with the connotation of reproducing/approximating physical laws but to be any form of virtually generated dynamic and rule-based process. *Synthetic reality* may be another alternative term. *Cyberspace* could be seen as another option, but carries the connotation of a unified and all-encompassing "aether" where a virtual environment is understood here to potentially be limited in its scale.
One might also try to differentiate between digital and virtual environments, both being generated by artificial means but the latter explicitly being a 3D reality familiar to physical reality concepts.

[8]Silver, J. (Producer) & Wachowski, L. and L. (Directors) (1999). *The Matrix* [Film]. United States: Warner Bros.

2.2.2 Mixed Reality & Environments

This section will take a closer look at what it means to mix realities and environments. To avoid confusion, most examples will be limited to considerations about mixing only two environments - the physical environment *PE* and the virtual environment *VE*. There may potentially be many different environments that are trying to mix of course, and in these cases they will receive distinguishable labels.

Expert opinions and interpretations

Next to VR and AR, Speicher et al.[8] also asked experts to give their interpretation and important aspects for MR. The experts were asked to categorize three (deliberately provocative) examples as either mixed reality or not: (1) listening to music; (2) Tilt Brush, a painting application utilizing stereoscopic video HMDs and handheld controller motion for brush strokes (3) Super Mario Bros.™ (SMB), a video game typically played on a TV screen with a gamepad.

Example: Listening to music
Listening to music was not an example of MR for most interviewees in [8].

Discussion:
When looking at ME, just like with the VR 360° video example there is little dynamics within the environment. Given a visual application that would be decidedly categorized as MR by everyone, the mere addition of audio playback via attached headphones would probably only considerably elevate the MR experience if it dynamically reacted to the listeners spatial pose. It would be interesting to see how experts would categorize a technology like the "AUUG Motion Synth"[9] where 3D hand/finger poses and movements dynamically adjust tone, pitch and octave to create live music but otherwise focuses on an auditive experience (the conductor's movement may actually be interpreted as visually enhancing to the experience). This may be an indication that dynamic (reciprocal) interaction (i.e. an exchange of events that have noticeable influence) from one environment to the other is key.

Example: Tilt Brush
The example of Tilt Brush was almost unanimously classified as VR instead of MR. One expert put it under MR with the reasoning that VR is a kind of MR according to the Milgram-Kishino continuum.

Discussion:
Following prior examples and argumentations on VR, for this work it would be more accurate to classify it as MR leaning heavily towards VR - but not exclusively VR. One may disagree with the subsumption of VR under MR based on the idea that - taken to its extreme end - a virtual reality does not need to

[9] `https://www.auug.com/` (last accessed 08-06-2022)

mix with physical reality. As for a VE, there is very little internal pro-active momentum on the virtual side e.g. the only dynamic comes from the user's inputs.

Example: Super Mario Bros.™
With the video game example of Super Mario Bros.™, there are a few interesting points as well. The experts of [8] were unanimous in their refusal to call SMB mixed reality. Reasons given were that it's just input, a missing spatial aspect, a notion of "if this is MR, then everything is" and a gap between real world and game GUI.

Discussion:
It is true for a single player video game, just as with Tilt Brush, that very little happens without the player's input. In comparison, a game that is essentially playing itself would be a sort of VE/VR within the interpretation of this work.

On the notion of input, arguably the press of a button on a gamepad is a more direct and faster link (and therefore integration) of a human's intention into a virtual environment than for example motion controls as is the case with Tilt Brush. It represents less physical effort (e.g. mapping physical actions to virtual reactions) and is therefore reducing the reach and complications of mixing both realities. The presentation of the game's output on a 2D screen however leaves a lot of possibilities to perceive physical reality and thus puts a lot of weight towards physical reality again.

Another interesting observation is that video games featuring a shared world (i.e. social aspects) with tradeable in-game resources - often located in the massively-multiplayer online (MMO) genre - regularly form mechanisms of exchanging real world currency for in-game resources. Some game publishers offer sanctioned ways for these transactions. But especially the cases where this happens without the publishers' intent or is even expressly forbidden by them as to not negatively affect the general player experience, it still happens quite often that a black market forms. Players want to exchange their real-world resources against virtual resources and would consider losing virtual resources a very real loss to themselves. The notion of "realness" seems to be heavily reinforced through the social aspects as well as all players having autonomy, reactivity and pro-activeness within the game world to some degree.
Yet in the case of SMB very few people would consider paying money to achieve something inside the game world to be a sensible course of action. This is probably due to the game's sole focus on the single player (no social aspect) that makes it entirely inconsequential after turning it off.

Some works further categorize additional realities with technology and mechanisms of extending, diminishing or altering perception of physical and virtual objects.
Schnabel et al.[17] categorize the methodology by which (visual) perception is

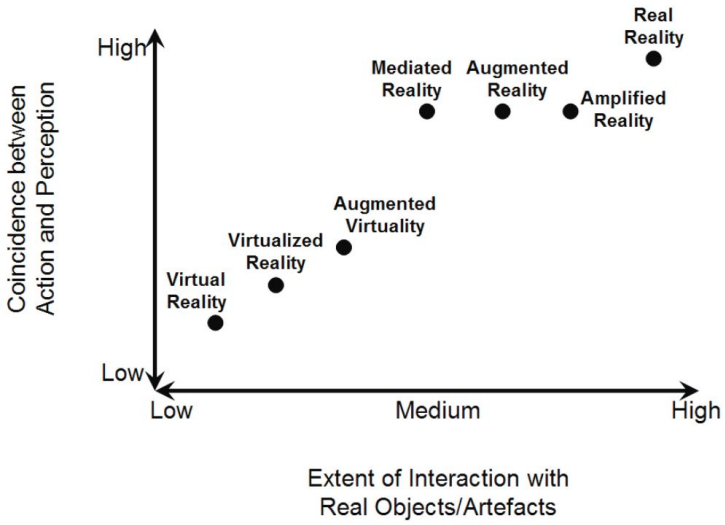


Figure 3: Classification of reality technologies according to Schnabel et al.[17]

altered as follows: *Virtual Reality* is presented as technology that "[...] creates a total Virtual Environment (VE) [...]". *Mixed Reality* in comparison is named as a more expansive form of VR with two major modes, *Augmented Reality* and *Augmented Virtuality* with one end of the spectrum being augmented/complemented by objects from the other end. *Mediated reality*, then, is a general concept of altering sensory input by computationally filtering it - this includes *Diminished Reality* which deliberately removes or blends out objects. *Amplified Reality*, on the other hand, is used to describe processes of objects controlling how inherent perceivable properties are being expressed. *Amplified Reality* is contrasted with *Augmented Reality* which super-imposes virtual information onto physical objects while leaving the perception of properties of physical objects (mostly) untouched. *Virtualized Reality* describes the process of capturing a physical scene in its spatial structure and virtualizing it, in turn being able to synthesize new virtual perspectives from that scene description.
They continue to place each term along two dimensions (see Figure 3):
1) The correlation of action and perception, meaning how much action and perception are happening within the same space. Handling physical objects, action and perception coincide and "we can see what we do".
2) The extent to which physical entities are interacted with.

As discussion on the term *Mediated Reality*, it should be considered that even the human senses and consequent signal processing in the brain are already a mediating step of constructing reality[9], as disorders like (visual) neglect show[18].

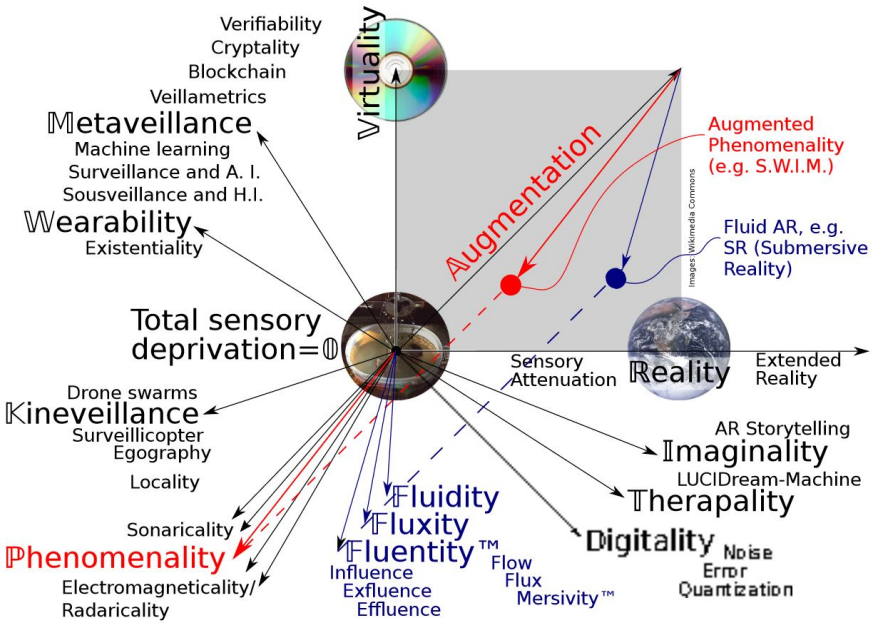


Figure 4: The Multimediated Reality Continuum according to Mann et al.[19]. Some axes are correlated, thus do not form an orthonormal basis.

Mann et al.[19] add to the discussion by investigating the term *XR* (or *X-Reality*), again presenting three conflicting interpretations. They give mediality its own dimension for categorization and provide examples for phenomenological aspects of altering reality perception that stay purely within physical means like making electromagnetism perceivable through visual or tactile means - calling it *Phenomenological Reality*.
They also discuss the role of *Artificial Intelligence* (AI) and *Humanistic Intelligence* as the concept of human-in-the-loop-AI. This puts the senses and effectors of humans in a feedback loop of controllability (surveillance) and observability (sousveillance) with machine sensors and actuators. This reciprocal sensing is seen as the fundamental technological basis for *Multimediated Reality*.
Multimediated Reality (Figure 4) is characterized as a multi-scale, multi-modal, multi-sensory, multi-veillant, multi-disciplinary and multi-dimensional continuum for describing "reality" technology. The origin of this multi-dimensional reality continuum is regarded to constitute an absence of any sensory stimulation (e.g. total sensory deprivation). *Multimediated Reality* is also about how humans interact with their environment, and technology extending their body and mind to that end. As an abbreviation for *Multimediated Reality* "All R", "*R" or "ZR" are suggested.

A conclusion from all of this could be the following:

Humans strive to alter/enhance their perception and agency through technological means. Whether it be our base biological senses or technology mapping to those senses, all of it serves as a mediating step to form our (subjective) reality of an (objective and independently existing) environment.

2.2.3 Ubiquitous Computing

Weiser [20][21] originally described ubiquitous computing as a world where information and computation is available everywhere in a world of fully connected devices which seamlessly integrate into everyday life while becoming virtually or effectively invisible. This is also described as "calm" technology existing and working peripherally [22][23], "invisibile" computing [24]. Comparing it against virtual reality he said that "Unlike virtual reality, ubiquitous computing will integrate information displays into the everyday physical world. Its proponents value the nuances of the real world and aim only to augment them" ([21], p.71).

Lyytinen and Yoo[25] provide a further distinction along the dimensions of mobility/embeddedness between mobile (high mobility, low embeddedness) and pervasive (low mobility, high embeddedness) computing. Ubiquitous computing will then combine the advances of both ideas.
Mobile computing can be achieved either by small-scale devices that can be carried, fit into pockets and clothing or by making computation available through broadband networks accessed through lightweight devices. The ability to bring and access computation almost anywhere on the world, however, also discloses a limitation when the mobile device cannot retrieve contextual information about its current working environment and adjust its computation accordingly. It is also stated that a manual reconfiguration to changing environments by the user is something that the average user does not want to perform.
Pervasive computing as the idea of an area being populated by sensors, pads, badges and physical/virtual models of physical/social/cognitive environments can remedy this lack of context by obtaining and providing information about the environment where the device is embedded in. The authors describe this process to be reciprocal in that the environment should also be capable of detecting other (mobile) devices entering the environment and making appropriate changes in their computation.
This mutual dependency and interaction is estimated to lead to a new capacity for systems to act more "intelligently" upon and within their environment.

Different authors have described several characteristic features of ubiquitous systems and applications. These give an overview and can inform technical decisions as well as challenges a system is potentially facing:

Distributed, Omnipresent Systems
The environment will consist of many small-scale mobile and pervasive devices located in very different places, relying on wired and mostly wireless inter-

connectivity for communication. The networking between devices will likely be decentralized and modular. [21][26][27]

Heterogeneous Devices and Information
The devices in use may be personal or shared in nature. They greatly differ in capabilities (older generation devices, cheap vs. expensive, manufacturer, different nature/purpose, etc.) and purpose (efficient / proprietary data formats). Multiple services including different databases, data formats and access rights are involved in a single situation. Applications themselves also have to move between devices when engaged at different times or circumstances which requires interoperability. [28][27][29]

Appearance, Explicitness
Devices and computation manifest themselves in a calm, invisible, disappearing, peripheral, ambient background manner. Computational capabilities are embedded in everyday objects and situations. [21][22][23][24]

Resource Constraints
Small-scale devices have limited computational power, the physical size also affects e.g. available display space. Since devices can be mobile, the overall available resources also depend on the currently present amount/type of devices. Available bandwidth is also limited with increased number of devices and increased communication. [21][28][25][30]

Context-Awareness
A central aspect of making computation smarter is to enable devices to share their context and work within the context of their spatial, functional but also semantic surroundings, thereby adapting computation to changing situations in order to reach better performance and/or more helpful conclusions. It also opens avenues for completely new possibilities and considerations for human-computer-interaction when input can be given indirectly and effortlessly. It is a foundation to better support heterogeneous device constellations, application/task adaptation and resource constraints.
[25][26][31][32][27][33][28][34][35][36][37][38][39][40][30]
This topic will be elaborated in Section 4.7.

Dynamic Adaptation
Environments in general but also user's intentions, goals and actions change due to new developments and circumstances. Applications will (automatically) adjust accordingly, and communicate what restructurings happened and why. [28][26][36][30]

Social Factors
Ubiquitous scenarios will in most cases involve a social component. The social environment of a user plays a role in terms of context. But the technology itself also has an impact on social dynamics. [28][27][26]

The Internet of Things (IoT) as a highly related field has been quoted as a technology with similar characteristics: Large Scale, Intelligence, Sensing, Complex System, Dynamic Environment, Massive Amount of Data, Heterogeneity, Limited Energy, Connectivity, Self-configuring, Unique Identity, Context awareness. [41]

With this vision of course also come challenges and novel considerations. Greenfield's book titled "Everyware: The dawning age of ubiquitous computing" [40] discusses many technical but also social aspects.
In terms of individual user experience these are some of the topics identified:
The user experience will be permanently engaged instead of an intentional and focused human-machine interaction ([40] Thesis 9). Users also get engaged inadvertently, unknowingly and unwillingly ([40] Thesis 16). One has to consider that most users will not be experts in this kind of technology ([40] Thesis 17). The use of such systems is ambient and peripheral instead of active and focused ([40] Thesis 18). Such "calm" technology then carries the problem that the physical form of objects may not hint at the digital capabilities that are attached ([40] Thesis 38).
The new and indirect ways of interaction with the digital environment will also require new forms of HCI. Haptic interfaces in the form of tangible media or physical computing are mentioned as interfaces that join the physical and digital spheres and go beyond what we already know from e.g. voice and touch interfaces. Interfaces that understand natural gestures can work intuitively and effortlessly, in more complicated cases they can rely on learned movements and muscle-memory of the user. Some gestures are culturally specific or different though, another point to consider for context. ([40] Thesis 10)

One should also take into account that ubiquitous systems can encompass all scales and varieties of locations.
This ranges from the human body (biosensors) to rooms, buildings, cities, and the globe ([40] Thesis 11-15). On larger scales, dealing with a multiplicity of users, inputs, intentions, devices and systems may lead to conflicts ([40] Thesis 20). Yet this (re-)combination of practices and technologies may also grow to become bigger than the sum of their parts ([40] Thesis 21).
Weiser[21] includes a quick estimation of how bandwidth intensive such an environment would be, even for just an office building and further suggests that the crucial metric should be *bits per second per cubic meter* - indicating that respecting locality of data should make a big difference in bandwidth consumption.

There are some potential and some definitive risks that have to be paid attention to.
The inter-related nature of everything is something that could be taken advantage of. The highly enriched and available amount of data carries the risk of being cross-correlated if not secured and protected, i.e. an attacker inserting itself into the loop and accessing and recording data not meant for them could

lead to malicious data mining and unwanted identification of users across applications and environments leading to extensive profiling and transparency of human beings ([40] Thesis 22).
Systems not only need to protect against malicious abuse, they also need to consider social impact and implications, how cultural differences are reflected in the internal decision-making, semantic reasoning and context analysis ([40] Thesis 23). The introduction of such technology, especially with respect to before unrelated and untouched everyday situations and analog elements, can be intrusive, mediating and transformative to the original process ([40] Thesis 34).
The exposure of otherwise latent information that would typically go unnoticed or is quickly forgotten is at risk of being made explicit or even recorded by ubiquitous systems, which can lead to social and psychological discomfort ([40] Thesis 35). The goal of making all aspects of daily interaction machine readable and understandable also means that they have to fit into technical standards, formats and protocols. This, too, is a certain kind of explicit exposure and imposes a "rigid" approach to things as it can influence the way humans think about such systems when everything needs to be identifiable, searchable and fitting into conventions ([40] Thesis 36). There is a certain contradiction in many elements of everyday life being tacit, unprecise or unspoken versus IT systems that need things to be as explicit as possible ([40] Thesis 37).
When talking about "calm" and "hidden" technology, this also means that certain intentions or agendas behind their use are equally opaque to the user, making an informed decision about how their data is utilized difficult ([40] Thesis 39).
The seamless integration of so many aspects of life may actually be something that is undesirable. It may lead to a diffusion of boundaries, e.g. ownership and responsibility. This can contradict the initial desire for effortlessness in the interaction ([40] Thesis 40).

Some risks are also inherent to the technological architecture chosen to realize ubiquitous systems.
The freedom given to users also depends on the architecture and design of a system to some extent ([40] Thesis 42). The algorithms, standards and norms will require human agency, judgement and compliance in one form or another ([40] Thesis 43).
There is a general tendency in humans to anthropomorphise and socialize interaction with artificial systems. For that reason it may be incumbent on ubiquitous systems to become more "friendly" and relatable in appearance and behaviour in order to be received more generously and less like a nuisance ([40] Thesis 45). When a user is faced with new, unfamiliar, advanced technology or considers themselves less adept than desirable in handling it, there's the impulse to blame themselves for faulty behaviour and interaction. The system is rather accepted as given, good and working as intended even if this may not be the case ([40] Thesis 46).

Finally, some challenges for conducting research can be given.

Ubiquitous computing as a field in its early stages of development involves not only solving existing problems. Instead it will encounter issues worth investigating that are still to come up through new inventions, conceptions and imaginations. [25]
It will involve very individual problems on a global scale, e.g. effects of wearables on a global market in various environments. [25]
Research must involve both social and technical aspects, and analysis must be conducted on different scales from individual to teams to organizations. [25]
To achieve adaptable and composable environments a semantic modeling is required. This includes models for user preferences, tasks, goals, needs and device capabilities. This can be achieved through ontologies. [28][42][43][44][45]
A software infrastructure has to be built that is capable of finding, delivering and adapting relevant applications for a given context, environment or task [28][46]
Applications need to be developed and configured according to their composition of services and user interfaces. Their data flow needs to be orchestrated. Applications will be seen more as a high-level description of a user task instead of a single piece of software targeted towards a particular environment/hardware. Applications will also need to specify their interaction logic on the level of intentions and their requirements on the basis of data and computation. Creating reusable services will be a bigger challenge as they will relate to physical artifacts (sensors/actuators) for a particular case. [28]
Finally, validating user experience will be an extensive task, consisting of large-scale user studies and wider deployment. The scenarios are inherently dynamic with varying social backgrounds and use of heterogeneous devices. This makes conducting consistent evaluations more complicated. It will involve more field research instead of contained laboratory experiments. [28]

Relevant fields of application are manifold. [27][26]
Identified, among others, are mobility[34][47] & logistics[48][49] , living[50][51], health care [38][33][52], industry and economy, food and pets, security of documents, ticketing, education [39][37][53], workspace, travel, entertainment and military[54].

2.2.4 Reality vs. Environments

Aggregating all the previous considerations in this section on the nature of physical reality, mixed reality, virtual reality, their environments and how they intertwine, a conclusion relevant to this work should be made about an appropriate terminology to represent realities, environments and what is necessary to allow them to properly mix and interact.

The taxonomy of [17][19] are helpful additions to clearly distinguish the different approaches of altering perception and presenting virtual content, thus the word *reality* is entirely fitting.

This work is more concerned with the environments and clearly distinguishes them from the technology used to represent them as a reality. There should be a clearer split between the technology used to "access" virtual content - i.e. the mediating step that allows perception - and the virtual content itself. As soon as MR applications evolve beyond a single technology focus, as soon as they become more than a closed sandbox entirely dependant on the processing hardware, the environment itself will play a distinguished role and must be recognized as a separate problem from the reality technology. The content or environment may in the future not be accessed through the same means by everyone, meaning one user may opt to view it on a 2D desktop and another user may put on a VR HMD.

When it comes to the term "reality", there is certainly a very deep philosophical or ontological rabbit hole that one could descend into. In this context, radical constructivism[9] should only be named as a school of thought that makes a clear distinction between the objective reality or environment and the subjectively constructed interpretation of the same reality. This is not the focus of this work however.
Instead this thesis is trying to find some very base assumptions of what constitutes an environment, how these base principles can be used to describe different environments (physical and virtual) and how to support dynamic behaviour between them. The goal is to derive a consistent approach to technically describe exchanges between environments and how to effectively model them:

1) The first assumption is that an environment consists of (more or less) persistent entities with certain properties. The properties can be expressed in some form, and can possibly be influenced through processes inside and outside of the entity. It also means that different observers can come to similar conclusions about the properties or state of entities and the environment in general. This consent often reassures its "realness".
2) The second assumption is that interactions or events between entities follow dynamic but consistent laws of causality. They imply relations and processes. The reactions to these actions/events, too, must have a certain persistence and consequence. A system where actions do not have perceivable, lasting, relevant and comprehensible consequences is difficult to be accepted as "real".

Addendum:
Something that goes along with the base assumptions but only plays a role when relating environments to each other is a demand for consequence and agency. If the processes within one environment A can not have noticeable consequences for a second environment B beyond a minimal fleeting scope or single individual observer, the first environment A will usually not be considered "real" by the generality of the second environment B. If neither environment can affect the other one in a meaningful way, there is little need to pay attention to the events of either from the perspective of the other one as it may just as well not exist or be an internal part of a singular entity. Nevertheless, to entities within one

environment it is still "real".

It is a bit unfortunate and possibly the source for some of the divergent interpretations found in related work presented that these assumptions very much fit a description of objective reality. But as the term virtual reality is loaded in research already, it seems to invite even more confusion to talk about (object) physical reality and virtual reality in this context. Instead the terms "physical environment" and "virtual environment" are used to describe the objective base here and anything carrying the suffix "... reality" is taken as any form of subjective interpretation or mediation of the former. As such, the term physical environment could be exchanged with objective physical reality but to form a clearer distinction while reading this work and because it has some fitting analogous use in related fields of science, the term environment will be preferred.
For physical environments then, these two assumptions are obviously reflected in particles and the laws governing forces and interaction between them. For a virtual environment, it may try to simulate physical laws in their detail and operation yet it may also be very abstract, simple and/or artificial in its function - since one of the enticing aspects of VR and MR is often stated to be the ability to diverge from or expand on what would physically be allowed.

3 Re-Embodiment Scenarios

This chapter describes the original idea from the introduction about re-embodiment in VR. It first explores some of the works related to theoretical background about embodiment and how it applies to VR and MR applications. After that, it describes some of the experimental setups that happened before the development of Ubi-Interact - indicating some of the considerations that later influenced its design. It closes with a setup testing Ubi-Interact's applicability and intended modularity for embodiment scenarios specifically and MR scenarios in general.

3.1 Related Work

The examples and scenarios native to ubiquitous mixed reality deal with a close or sometimes even intimate relation between technology and its user by wearing devices that directly overlay our senses or monitor our physical status (HMDs, digital glasses, spatial audio, biosensors, gesture/posture recognition, EMG, EEG, ...). Human cognition is also targeted for integration, sometimes explicitly through e.g. EEG devices, sometimes more implicitly.

VR technology tries to "drag" a user into a different world and offer new experiences. Quite often the stated goal is a feeling of immersion and presence for the user. How the user is represented inside the virtual environment varies: they may be a silent watcher - incorporeal but able to look or move around, they may be granted (usually handheld) gadgets through which they can trigger specialized interactions or they may receive a full body representation that moves in sync with their physical body.

With ubiquitous computing, the question is more about how devices and virtual interfaces influence and integrate with ourselves, our body and the image we have of ourselves when they are always present and are physically close to us and/or easily reached in the virtual sense - similar to how we use our smartphones today already.

Whenever there is an effort to bring differing environments together there is the issue of how to represent an entity from one environment by means of another environment. If the environments are dissimilar enough they appear as entirely abstract, indiscernible or unknowable to each other. An entity thus needs a form and expression that fits the other environment for it to be embedded and recognizable. Examples for this are the virtual replacement body in VR but also a ubiquitous service making itself aware and engaging with us.
At the same time, the form an entity takes may very well influence the way it interacts within its environment's constraints and how the internal computation or cognition attunes to these interactions - in that sense outer form/representation also influences internal processes. This in turn can have a reciprocal effect on

the entity itself and its behaviour. [55][56]

This concept of external form affecting internal or underlying computation/cognition, forming a more and more inseparable and indiscernible unit the tighter this integration becomes is represented in several aspects of this work: whether it be the virtual self representation of a human in egocentric VR, the use of additional tools and interaction devices in a ubiquitous digital environment or the steering of co-located, remote or virtual robotics systems.

3.1.1 Body, Mind, Tools and Technology

We use technology as a tool to enhance/alter/substitute our capabilities. In the case of physical (motor/sensory) capabilities, tools can be the carpenter's hammer, a pen for writing, glasses to improve vision, the sword of a fighter, a musician's instrument or the keyboard used to type out this sentence.
Neurological studies[57] suggest that during tool use our (visual) perception does not stay limited to the hand wielding the tool but extends to the contact point or target of the tool where its intended effect actually occurs. During use and with increased proficiency, the tool itself takes a background role in our perception and becomes, in a sense, quasi-transparent. One does not have to look at the keyboard while typing, the fingers know where they are placed and where to move. And if one were to exchange the keyboard that one is used to with a differently-sized or formed one, one would loose some of the typing speed and would need to recalibrate first. Likewise, the blind person's cane and its mediated tactile feedback to the hand is so well integrated that it can be used as a "feeler" for the environment.

Furthermore, technology can also expand cognitive capabilities. A prominent example given by Clark and Chalmers[58] deals with a person suffering from Alzheimer's disease who is using a notebook to remember crucial things and navigate the world. They argue that the constant use and heavy reliance on their notebook makes it essential to their life and to them as a person as a base for their cognitive processes much like a healthy person relies on their memory. It plays as much of an important role on the cognitive level as the blind person's cane becoming an essential "feeler" to them on a physical level.
The smartphone we use daily is another highly personalized device assisting in communication and memory.

3.1.2 Body Image & Body Schema

The automatic bottom-up sensory and organizational integration process to form an internal representation of the body is called a body schema. It is dynamic and malleable to some extend, representing spatial and biomechanical

information derived from multi-sensory input that can also include important aspects of the environment, even just temporarily. It is not limited to the biological body, but may include tools and technological extensions [59][60].
Complementary, the top-down representation is referred to as body image. It relies on lexical-semantic descriptions and is connected to three modalities: 1) perceptual experience of the body, 2) conceptual understanding of the body and 3) emotional attitude towards the body.
Body schema and image are not separate. Examples stretching into both image and schema and exploring their connections are the cane of a blind person ([61], p.165) or a prosthetic limb ([62]). [63][64][65]

When it comes to VE and ME applications, it is clear that the goal is to have objects/tools/interfaces extending into the different environment and allowing us to incorporate them into our body schema and use them to become an active participant.
The same could also be said for truly ubiquitous computing environments. In an environment surrounded by virtual assets, humans are generally blind to what is happening around them on the other end of the spectrum. We need either an equivalent to the blind person's cane that we can use to "get in touch" with it - some of these interfaces eventually becoming part of our body schema again. Or the virtual systems need some way of making themselves known to us. A ubiquitous service accompanying us as users may have multiple ways of embodying itself on displays, through audio alone, etc. depending on the situation. It may be beneficial to them to rely on a variable "body schema" for their smart assistance as well, relating back to some of the points made by Greenfield[40] about "friendly" and relatable interfaces.

3.1.3 Disembodiment vs Re-Embodiment

Especially in the case of VR there often exists the notion that - in a cartesian sense - the body is left behind and it is only our mind and our senses fed by the VR technology that are important to consider for the virtual world.

Merleau-Ponty however argues from a phenomenological point that there is no such reduction. The body and mind can not be mutually separated ([61], p.84-102). There is further neurophysiological evidence that such a separation is incomplete and that embodiment is a bottom-up as well as a top-down process. [9][63]

In comparison, true disembodiment occurs during a failure of body schema as well as body image to induce desired motor skills. It is the feeling of an alien object that one has no control over.

De Preester[62], too, argues that the use of technology does not imply the human capability for *disembodiment* but instead requires *re-embodiment*. We

do not so much leave our body behind, instead we extend our bodily self to include the technology, even just temporarily.
De Preester further offers a distinction between body extension and incorporation of non-bodily items, the latter involving more drastic alterations and conditions. Tool use, for example, is seen as changing motor and sensory capacities, but not the feeling of body ownership.
True re-embodiment is then seen as a matter of incorporation with a required shift in subjective experience. In a categorization of limb, perceptual and cognitive extensions/prostheses this would necessitate a changed feeling of body ownership, a change in subjective perceptual experience and a change in felt ownership of thought respectively.

Finally, Buongiorno[66] argues that for the digital space, the distinctions made by De Preester are more complex and that digital embodiment is a blurred complementary process of bodily extension and incorporation of objects.

3.1.4 Virtual (Re-)Embodiment

Kilteni et al.[67] built a working definition for the sense of embodiment in VR. They identify three underlying supporting senses: self-location, agency and body ownership.
Self-location refers to the spatial experience of being collocated inside a body. Out-of-body experiences constitute a disturbed self-location. [68][69]
Agency means motor control that follows intent and the concious experience of will. [70][71][72]
A sense of body ownership is connected to a feeling of possession over a body and the body being the source of sensory feedback [70][71][72]

Concerning fundamental limitations of virtual re-embodiment and telepresence, Dolezal[73] gives the impossibility for risk of bodily harm (death in the extreme case). Physical contact and proximity are also mentioned as cases where virtually replicating e.g. the comfort of touch or presence inherent to physical reality seems impossible.

As mentioned by Tsakiris et al.[74], synchronicity between multi-sensory feedback (e.g. visual and proprioceptive) is a crucial factor for embodiment. The increasing end-to-end latency between the user's movement and visual feedback through the VR HMD will negatively affect embodiment efforts.
Caserman et al.[75] cite a number of thresholds evaluated through user studies. Results show that a latency beyond 63ms for visual updates induces cybersickness symptoms. Above 101ms feeling of body ownership was concluded to be significantly affected.
A study by Waltemate et al.[76] cites 75ms as a limit to motor performance and perception of simultaneity. Sense of agency and body ownership are determined to deteriorate at thresholds of 125ms and again noticeably above 300ms but

never vanish completely.
The effects of linear latency vs. jitter were compared by Roth & Latoschik[77].

Seeking a validated method of measurement for virtual embodiment, Roth & Latoschik[77] designed and tested a virtual embodiment questionnaire.
They highlight the three factors of ownership, agency and change in a perceived body schema.

VR opens the possibility to display the environment from a non-egocentric perspective. Different studies[78][79][80] have tested the effects and came to the conclusion that diverging from an ego-centric reference frame and visual perspective overall negatively affects embodiment.

VR not only allows a change in perspective, but also of body structure and capabilities though.
Kilteni et al.[59] investigated to what degree a virtual arm could diverge from the physical one in posture and length before breaking with the sense of ownership.
Kasahara et al.[81] performed a study on how visualizing one's body in VR in a spatio-temporally deformed state affects embodiment and the perception of one's body. They came to the conclusion that perception can be altered, e.g. extrapolating movement by around 100ms into the future would cause a feeling of reduced weight.

As Yee et al. [82] argue, VR can also be used an escape to the "tyranny" of the laws governing us, inventing beneficial ways to partly break with or subvert familiar mechanisms of embodiment like being able to keep eye contact with more than one person. Such departures from the conventions and limitations of our physical experience may provide beneficial or simply interesting experiences.

3.1.5 Robotics, Teleoperation, Telepresence

The fields of robotics, teleoperation and telepresence also offer many examples of control over additional or remote limbs and machines. Often, the goal is to recover or increase (felt) agency while keeping the cognitive load low. The system's design should support the user in both directions.

Llorens-Bonilla et al.[83] designed a Supernumerary Robotic Limbs (SRL) system with two additional robot arms worn as a backpack. It is meant to reduce workload, especially in over-head tasks, by assisting with grasping and holding objects.
They emphasize the importance of communication and coordination between the SRL system and its operator. Decoding the intention of the user is a key aspect for the system to work properly.

Penaloza and Nishio[84] performed an experiment where healthy participants

were challenged to control a third robotic arm via a Brain-Machine-Interface (BMI) using non-invasive EEG sensors.
The study compared performance between a single task condition of grasping a bottle with the robot arm and a multi-tasking condition adding a ball balancing task using the subjects' natural hands in addition to the robot arm grasping.

Martens et al.[85], too, utilized a non-invasive EEG interface to control a robot. They conclude that the system could potentially be improved in its awareness of incorrectly interpreted commands by including and learning from error-related potentials[86] detected via the BMI.

Teleoperating a Nao humanoid robot has been achieved by Stanton et al.[87]. To train the mapping of movements between human and robot body, they utilized full-body motion capture and machine learning.
During a calibration phase, the human user was asked to follow a pre-programmed sequence of movements and perform them in synchrony with the robot. A neural network was then trained on the synchronized data of body poses between human and robot. The trained network would then map the human motions to the robot during live use.

Sivakumar et al.[88] used available online videos of hand movement to train a teleoperation system on the task of translating human hand movement captured with a single camera onto a robot arm with a four-fingered hand. The tasks for previously untrained subjects involved object pickup, rotation, stacking, pouring, opening and closing drawers.

3.2 Libraries, Tools & Platforms

Development of the systems described hereafter naturally involved additional software. This is a short list of the major ones and a description of their specific relevance.

3.2.1 Unity3D

Unity3D[10] is a game engine providing visual rendering, physics simulation, animation, GUI, I/O and component-based scripting APIs. It also provides integration for VR and AR applications. It enjoys widespread adoption in the scientific community.

[10] `https://unity.com/` (last accessed 26-03-2022)

3.2.2 Three.js

Three.js[11] is a cross-browser 3D library based primarily on WebGL rendering. Animation modules are available and it can be integrated with physics engines like *ammo.js* or *Oimo.js*.

3.2.3 babylon.js

Babylon.js[12] is an open web rendering engine.
Some of the relevant features are integration for WebXR and the Mixed Reality Toolkit library as well as animation and physics engine support.

3.2.4 Gazebo

Gazebo[13][89] is an open-source robotics simulation platform. As such, it focuses on accurate sensor and physics simulation but also provides visual rendering. A transport module provides a pub/sub messaging system and services which build upon *Google Protocol buffers* and *ZeroMQ*.

3.2.5 Nengo

Nengo[14][90] is a Python-based library for spiking and non-spiking neural networks.
It features an easy to use GUI and special API allowing to map mathematical functions into SNNs.

3.2.6 Neurorobotics Platform

The Neurorobotics Platform (NRP)[15][91] is a platform connecting different simulators for the purpose of experiments embodying brain models in simulated robots. It features brain or neural network simulators that can be connected to robotics and physics simulators.

The mission statement of the Neurorobotics Platform[91] is to investigate biologically realistic models of the brain via spiking neural network simulations.

[11] `https://threejs.org/` (last accessed 26-03-2022)
[12] `https://www.babylonjs.com/` (last accessed 26-03-2022)
[13] `https://gazebosim.org/home` (last accessed 26-03-2022)
[14] `https://www.nengo.ai/` (last accessed 26-03-2022)
[15] `https://neurorobotics.net`

In order for such brains to perform realistic sensory-motor tasks, they require *embodiment* in a realistic environment. They require a robot body that anchors them. The physical construction and properties of this robot, e.g. the length and weight of a gripper arm or the material it is made of, do need to be considered and attuned to by the brain to use them efficiently. A change in the length of the robot's legs, their material's flexibility or their contact surfaces requires a change in the computational processes inside the brain that make the robot walk properly.

In its current version (up to 3.2) the supported neural simulators are NEST[92], Nengo and TensorFlow[16][93]. For robotics and environment simulation, Gazebo and ROS are deployed.
All simulators are updated in a closed loop to guarantee their synchronization. In order to transmit data between the different simulators and their individual design architectures, data formats, etc. the NRP uses so-called *Transfer Functions*. These are Python code snippets called between each simulator update. They are the means to map and transform data and signals from one simulator to the other, e.g. transforming spikes from brain simulator to voltage for a robot actuator.

With version 4.0, the architecture of the NRP becomes more open and general-purpose. Simulators are encapsuled into *Engines* and an arbitrary number of them can be connected into a synchronized simulation. The concept of *Transfer Functions*, now called *Transceiver Functions*, is kept.

The NRP also includes a web frontend, offering among other things a visual rendering through Three.js of the simulation environment whose updates are communicated via ROS topics.

3.2.7 PID Controller

PID controllers[94] are a form of continuous feedback loop control well established in industry and robotics.
They are commonly used to regulate the dynamics of a system, like a robot arm driven by a motor moving into a target position, with the goal to avoid overshooting the target while still providing a minimal or adequate response time for reaching the target.
The controller works by constantly calculating an error term between the set target state and the current state of a system. Based on this error the three name-giving proportional, integral and derivative correction terms are calculated and used to update the system state.
Each of the correction terms can be tuned by a factor, giving the adjustable parameter space for a PID controller. Variations like PD controllers with two

[16] `https://www.tensorflow.org/`

parameters also find application.

3.3 Dealing with Discrepancies between Environments

Whenever environments are brought together the degree to which they are spatially, temporally and/or semantically separate in their interaction is an influence to be respected.
Especially when both environments appear next to each other or are supposed to work similarly as with typical VR/MR applications and 3D spatial registration and tracking, the immersive integration of these realities can quickly form expectations about what must be happening inside the distinct environments even if their separated nature makes this impossible.

Some examples already given in Section 2.2.1 and 2.2.2 describe a clear mismatch of object placement and general structure between a physical and a virtual room when body movement is tracked and a one-to-one mapping between environments is desired. As seen in Section 3.1.4, introducing even a small amount of latency between the movement of the physical arm and the movement or just visual rendering of the virtual arm can result in disturbing the expected behaviour of the observed virtual arm which leads to a loss of embodiment.
In this case the issues may be prevented through increased performance or more detailed observations about either environment. At some point, however, discrepancies become unavoidable because the limitations like a wall or similar blocking object are inherent to either environment and not reproducible or intended to be reproduced on the other end. Here, different solutions have to be pursued.
If, for example, the virtual arm adheres to a simulation of physics, that same simulated realism may forbid it from moving exactly like the physical arm because of diverging properties like mass and weight distribution or motor strength and range of motion. Teleoperation scenarios show the exact same issues, only there the behaviour of the other environment is more evidently different and thus there is less demand or expectation of an alignment between both.
With increased immersion, any occurring mismatches are less expected and potentially disturb embodiment as studies cited in Section 3.1.4 show.

Together with Jonathan Haudenschild[17] a small-scale preliminary study on detected body-posture discrepancies between human and humanoid virtual avatar has been conducted. The study compared the effectiveness and induced level of discomfort of different methods for visual, auditory and haptic feedback to replace the missing tactile and proprioceptive feedback.
One of the scenarios was how to deal with physical impossibilities, e.g. when a

[17] Haudenschild, J. (2018). *Virtual Embodiment: Dealing with Discrepancies between the Virtual and the Real Body* [Unpublished Bachelor's Thesis]. TUM.

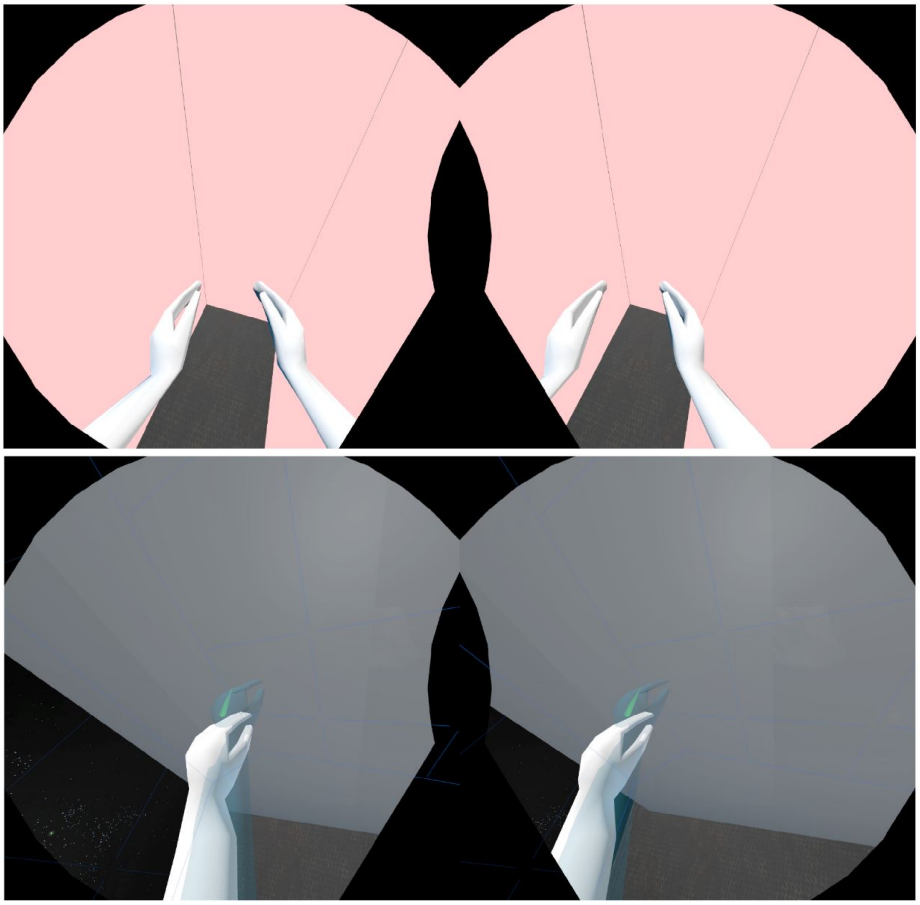


Figure 5: Top: User positioned inside a wall, view of outside geometry is blocked and unnatural color indicates undesired state. Bottom: ghostly second arm visualizing discrepancy between limb postures. Adopted from Jonathan Haudenschild's thesis.

user standing unobstructed in a room is free to move their head or whole body in any direction whereas the virtual avatar would be blocked by a virtual wall. As physically restricting the user's movement would require considerable additional effort in hardware and pose potential risks, other methods of providing feedback and preventing users from peeking through walls have to be investigated. Visual effects on their field of view like blurring, distortion, fading to black and changes in virtual geometry appearance were tested. Visual distortion received very negative feedback while blurring, fading to black and a change of colors were graded highest but also resulted in disorientation and discomfort for some. Blocking out all geometry apart from the interior of the wall upon entry would let subjects keep their orientation while also blocking their view (Figure 5, top) but was not as effective in preventing this behaviour as e.g. fading their vision to black.
Regarding discrepancies in limb posture when for example bumping into objects, three feedback methods were evaluated for the arms. Especially in cases where limbs are out of sight the lack of tactile and proprioceptive feedback is exemplified. Firstly, visual feedback in the form of a ghostly second representation of the user's own arm posture was made to appear once divergence between it and the avatar's arm posture exceeded a certain threshold (Figure 5, bottom). Secondly, auditive feedback in then form of a monotone sound was given. Thirdly, the controller held by the user and used to track their hand position was made to vibrate giving tactile feedback. Here, the tactile feedback was decidedly preferred.

3.4 Robot Hand Control using sEMG

Together with Tieck et al.[95], an experiment was conducted integrating the signals of a surface electromyography (sEMG) armband with a robot hand. It detected finger movements through the sEMG sensors and subsequently triggered movement reflexes in the form of motion generators for appropriate robot fingers.

The experimental setup (Figure 6) used the ThalmicLabs Myo Armband[18] and the Schunk SVH robot hand[19] with their signals connected via a spiking neural network (SNN) pipeline simulated in Nengo.
EMG data was accessed using an open-source Python API[20] while the Schunk hand provides a ROS interface. Nengo as a simulator is running in Python.

Figure 7 shows the full pipeline. Here, instead of focusing on the technical details and implementation steps it is more interesting to focus on the general segments and how they could be applicable to other studies and setups,

[18] `https://developerblog.myo.com/` (last accessed 08-06-2022)
[19] `https://schunk.com/de_en/gripping-systems/highlights/svh/` (last accessed 08-06-2022)
[20] `https://github.com/dzhu/myo-raw/` (last accessed 08-06-2022)

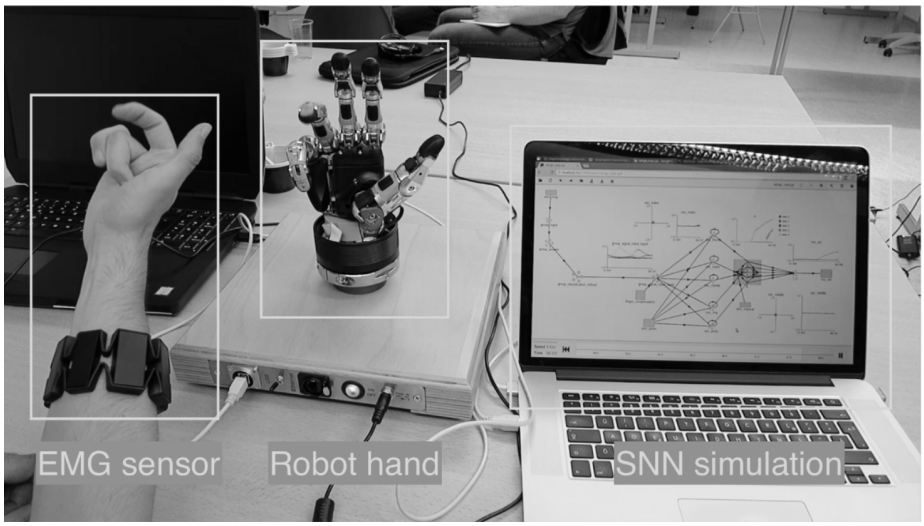


Figure 6: Experimental setup using the sEMG armband (left), the robot hand in mid-motion (middle) and the running SNN simulation using Nengo simulator (right).

especially since the whole setup was implemented before development on Ubi-Interact began and subsequently informed some of its design.
It starts with aquiring raw sEMG data via an API for the Myo armband. From this, the first group of Nengo networks generate a finger classification for one of the digits. This classification triggers a motion reflex primitive for the corresponding digit on the Schunk robot hand.

The Myo hardware used in this setup was at some point stopped for sale and support was discontinued[21]. The requirement of interoperability & extensibility cited in [96] echoes this issue.
This shows that experimental devices or small-scale production runs can potentially be phased out rather quickly or are iterated on with newer generations. Driver and SDK support is not guaranteed long-term so experimental setups are hard to maintain. Integrating such devices with newer software setups and versions can become difficult at some point. To counter this issue, it is beneficial to have an isolated and maintainable driver/SDK/API integration that can be combined quickly with other systems through simple communication interfaces.

Within the Nengo network, both segments of sEMG classification and motion reflexes can have separate uses and the pipeline as such can be divided into separate Nengo networks. The finger classification segment can find additional applications in novel user interfaces making use of individual finger movements using wearable devices. And while the network can be split, Nengo itself is a

[21] At the time of writing, the Myo armband is available for purchase again

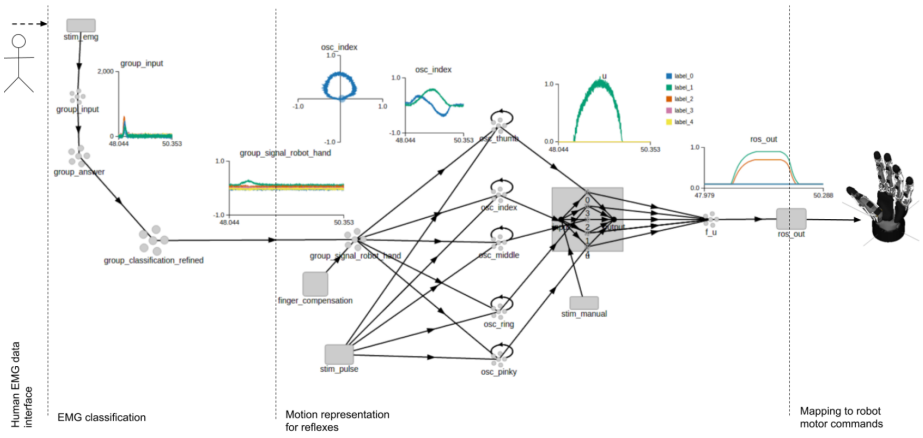


Figure 7: SNN pipeline with segments of EMG data interface, EMG classification, motion reflexes and robot motor commands.

dedicated software that can not be combined and integrated into just any other setup with ease.

The Schunk robot hand is a rather expensive piece of hardware that not every laboratory has at hand and may instead want to replace it or compare solutions with different setups. Models of the Schunk hand for virtual simulations are available and could replace some stages of experimentation and development.

As the results achieved in this experiment are certainly not final, it would also be of great value to have the setup retained in a flexible manner so as to exchange input methods, robot end-effectors or mapping algorithms between the two for continued study and improvements.

3.5 NRP Re-Embodiment

The NRP is a platform where embodiment experiments for virtual robots controlled by neural simulators in a simulated environment can be conducted. It is, of course, focused on the neurorobotics aspects of embodiment for learning and task performance.
If the same platform featured human agents in the environment, it would open the option for HRI (Human-Robot-Interaction) experiments. As more complex human behaviour and interaction is difficult to simulate, the only option is to represent human actors in the simulated environment. As such, they require an interface giving them enough agency to reflect various intricate behaviours.
The focus was on building an adequate technical foundation that would support embodiment experiments with respect to the conditions quoted in Section 3.1.

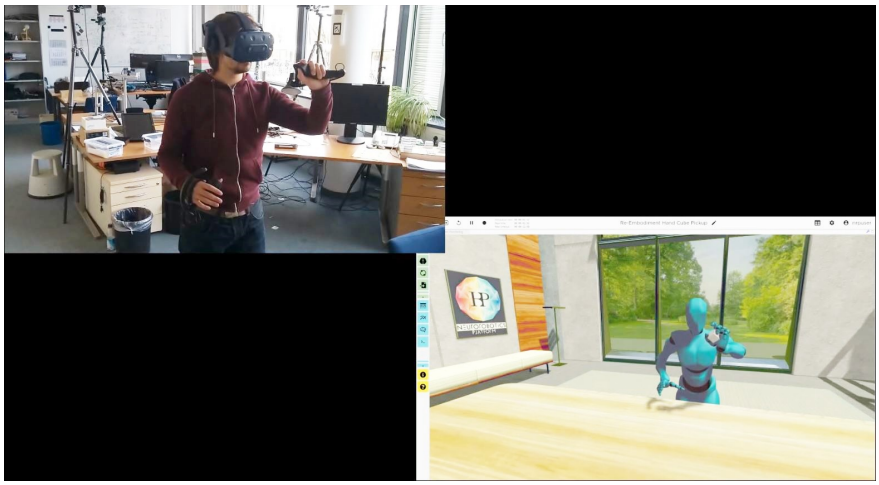


Figure 8: Side-by-side view of NRP setup. VR equipment with finger tracking (left) and simulation where user's hand motion translates to the robot avatar picking up a cube (right).

To provide this feature, the required additional elements are
1) a body tracking capturing the movement of human users,
2) a way of conveying the simulated environment to the users that is conducive to embodiment , i.e. a first-person VR perspective[97],
3) a body representation that is part of the environment simulation, i.e. a humanoid robot that the user can take ownership and control over and
4) motion control mechanisms translating the tracked human movements to the humanoid robot, i.e. 4a) determining a full body pose if not all parts of the body are tracked and 4b) estimating the forces necessary to move the robot avatar's body parts towards the body posture determined in 4a.

To get an impression of the eventual result, a video[22] is available. Figure 8 is a screenshot from this video showing an outside view of the VR equipment including finger tracking controllers and the performed grabbing motion by the user next to the simulation view of the avatar picking up a small cube from a table.

3.5.1 Unity VR Client

To cover the requirements of body tracking and visual representation, the choice fell to consumer level VR hardware and a software client implemented in Unity3D. Relying on widely available hardware would open the possibility for experiments involving humans to a much wider user base.

[22] `https://www.youtube.com/watch?v=-nTGG33ErNc`

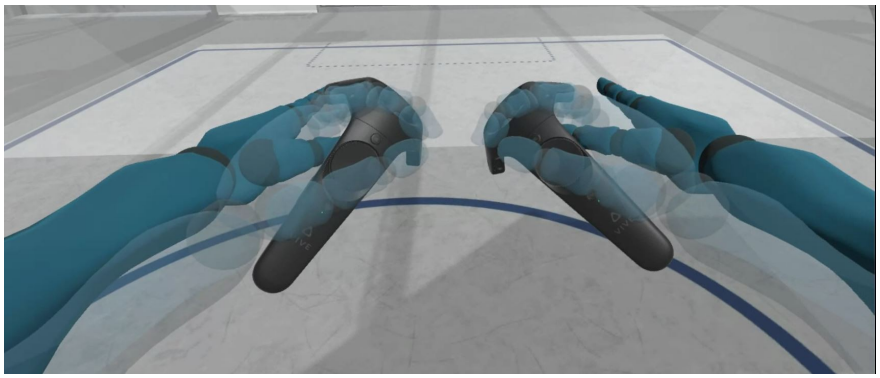


Figure 9: First-person view of robot arms. For illustrative purposes, controllers and estimated pose of arms (transparent) are shown as an overlay. Robot arms (opaque) are in the process of adjusting their position to the same as the transparent estimate.

Visual information and updates about the NRP environment are communicated via ROS topics and reflected in a Unity Scene. All of the physics calculations still happen within Gazebo.

Modern VR hardware supplies the necessary tracking for head and hands, with the option to extend to more detailed information like fingers, torso or feet tracking using additional or newer upcoming hardware developments.
To arrive at a full body pose from this, inverse kinematics (IK)[98][99] can give estimations of arm and leg poses. If no tracking data for torso or feet are available, IK targets can be approximated based on the head and arm positions and heights as well as a walking animation during movement.
Unity3D's character animation system already offers an API capable of a weighted interpolating between IK targets and animations. It was thus used to cover the above part of 4a for determining a body pose from the tracking data.

3.5.2 Humanoid Robot Avatar

As a physical representation model, the abstract X- (female) and Y-Bot (male) models available on Mixamo[23] were chosen.
The model geometry was then ported into the Gazebo native format of SDF[24].
This resulted in a model consisting of 83 individual rigid body parts connected by 82 joints.
Collision and inertia were added to body parts.
Joints were defined by their type and rotational constraints. The number of

[23] `https://www.mixamo.com/`
[24] `http://sdformat.org/`

joints results from some of the human joints that are movable in multiple dimensions (e.g. the shoulder) being represented by a concatenation of several one-dimensional robot joints to cover their respective degrees of freedom. The dynamic behaviour of joints was then handled through a dedicated plugin in a second step.

3.5.3 Motion Control Plugin & Limitations

The SDF file format has plugin library mechanisms by which different controllers can be applied to a model. Plugins allow initialization via a *Load()* function as well as iterative update calculations via *OnUpdate()*.
For the humanoid robot avatar, a plugin with several controllers offering different modes and aspects of motion control was implemented.

Regarding the control of joints, one of the controllers adds a PID controller for each point and ROS topics to set PID parameters as well as a target position for the respective joint.
Another controller opens topics to set a force that would constantly be applied during updates, offering a more direct low-level control without relying on a PID Controller implementation.
Controllers for direct application of linear and angular velocities and forces have also been implemented, controlling the robot avatar akin to a puppet with strings attached instead of forward kinematics using joint motors.

Apart from the motion control over limbs, there is also the question of how the avatar is supposed to be kept in an upright position.
The task of keeping a two-legged humanoid robot upright is in itself not easy and usually requires a purposeful design of robot legs, feet, certain limitations to its movement range and/or sophisticated software in order to keep balance [100]. These requirements are not reflected in the robot avatar that was chosen to be abstract but reflective of the full range of human motion capabilities.
User movements like leaning, stretching out legs, etc. would further complicate this task and require an elaborate sense of balance and control over foot placement and force contacts with the ground in order for the robot not to fall over immediately.
The hardware and input/output capabilities between human user and robot avatar in their current state do not offer enough fidelity to imagine such a detailed and dexterous control over a simulated body. The user is lacking any form of proprioceptive feedback regarding the robot limbs and it would require for new hardware to be designed first in order to convey this information in any meaningful way. It was thus decided to artificially support the robot avatar at its hips with forces keeping it upright / according to the tracked body pose at all times.

One of the aspired future experiments is to test how a motion controller

could consider both the user input and the required balance using mass distribution, velocities and torques to form a balancing pose control that merges both inputs. Artificial support could be gradually lowered to test the viability. This would in some sense resemble a physical rehabilitation experiment for the aspired balance controller.
This, again, is one of the challenges Ubi-Interact is trying to address - having personalized input/output from/to multiple directions merge and build an incrementally extendable system on it that can work in tandem with their user and extend their capabilities.

3.5.4 PID auto-tuning

With 82 joints and attached PID controllers that widely differ in their dynamics - e.g. a finger joint with grams of mass on each end being far down the forward kinematic chain of the body vs. a hip joint - a method for attuning the parameter space of each individual PID controller was necessary.
The latency of network communication and the future potential of personalized avatars with differing physiques are other factors that can influence the PID parametrization on a case-by-case basis.

Together with Markus Webel[25] - a tuning suite was developed that involved automatic as well as manual tuning options for on- and offline use. The tuning mechanisms were separated into modules with immediate visual and plotting feedback. This allowed to programmatically narrow down viable ranges of parameter space and then fine-tune the results manually until acceptable performance in overshooting and responsiveness were achieved.

3.5.5 Replicating humanoid motion

For the process of testing the viability and reactivity for human motions, relying on an actual user wearing VR hardware and performing movements or recording a full set of full-body motion captures was deemed inefficient. Instead the setup was extended to allow playback of freely available motion capture animations. The catalogue of animations included idle standing, walking, running, opening doors, jumping and even break-dancing. This also allowed to control the playback speed to test at what point parametrization of joint dynamics failed and/or the system would not be able to keep up anymore.

[25] Webel, M. (2019). *PID-Tuning Framework for Remotely Operated Humanoid Robots* [Unpublished Master's Thesis]. TUM. `https://wiki.tum.de/display/infar/MA%3A+PID-Tuning+Framework+for+Remotely+Operated+Humanoid+Robots`

3.5.6 Transition to Ubi-Interact

Due to the NRP's orientation towards deterministic and HPC simulations not running in real-time, it became apparent that further development of VR re-embodiment experiments with tight integration in the NRP ecosystem would not be feasible in the near future. The options were to continue with the existing setup relying on ROS, Gazebo and Unity or to embrace Ubiquitous Mixed Reality requirements completely and explore a system providing a flexible basis to adapt to the various device and simulation configurations with re-usable modules connecting them and taking care of the necessary intermediate transformations.

4 Towards building a Mixed Reality Framework

After a very general and conceptual exploration of how Ubiquitous Mixed Reality presents itself in Section 2, this chapter will take a closer look into the technologies involved that enable Ubiquitous Mixed Reality as well as related fields and how their influences can complement, inform and enhance solutions for Ubiquitous Mixed Reality.

Ubiquitous Mixed Reality encompasses quite a few (eco)systems and fields of research that play a role like the internet, robotics, IoT, machine learning, HCI and HRI.
In computation hardware it involves all levels from wearables to cloud computing. Traditional and prototypical devices are employed alongside each other to involve all human senses. They are intended for interaction within private homes as well as public places.
Many of them bring with them historically grown and established standards and protocols fitting their purposes. They formulate different demands for computational resources, data transfer and communication patterns, data formats, semantic representation and security among others. The following sections cover these aspects and present current practices.

This work focuses on the communication and discovery between devices and how to combine them in their capabilities - not just physical devices with buttons, etc. but also abstract objects of interaction.

4.1 Overview: Communication Patterns

One key issue for Ubiquitous Mixed Reality is the question of how to pass around data between all elements involved. A single standard of communication between them would be ideal but seems unlikely for the time being. Even with a future standard protocol established, what these fields of application involve, what context they operate under, how they integrate with their environment, their scope of connectivity, what data and formats they use and what is ideal for their efficiency - i.e. the meaning behind and interpretation of the data they use - is probably so diverse and needs to remain specialized to some extent that a generalized standard for data exchange between all of them seems rather implausible.

There are of course quite a few different communication patterns each with their respective advantages and disadvantages. Data distribution in Ubiquitous Mixed Reality scenarios are characterized by a very dynamic environment. It should be possible for elements to enter and leave at any point in time - maybe even move along a continuous "distance" measurement with the rest of the system being able to react.

All of this means that message distribution for Ubiquitous Mixed Reality must be thought of from two perspectives.
On one hand, when building a Ubiquitous Mixed Reality framework from the grounds up there is the question of how to pass messages within the system - between each part that is built with the help of the framework. Here a choice can be made on the basis of what patterns fit Ubiquitous Mixed Reality and the purpose of the framework best.
On the other hand, there are a lot of systems already out there with established messaging interfaces that belong in the bigger picture of Ubiquitous Mixed Reality. Simply ignoring them will not help bring about Ubiquitous Mixed Reality, instead the goal should be to find ways of communicating with them.
There are many existing examples of plugins and bridge components that are capable of translating between major and related development platforms and environments. This raises the question for a Ubiquitous Mixed Reality framework of how existing external and especially upcoming messaging systems can be inter-communicated with. If data is shared between two systems only, a direct one-to-one bridge between the two is perfectly fine. If however Ubiquitous Mixed Reality scenarios grow bigger and relevant context or computation involves data from three or more participants, a separated implementation being able to disseminate between all participants starts to make sense.

For Ubiquitous Mixed Reality scenarios, we have a few conditions that should inform a decision. Depending on the scenario, some communication patterns fit requirements of bandwidth use, update frequency, scalability, adaptability & plasticity, etc.[96] better than others. Here are presented three major communication patterns and the use-cases they are best suited for.

4.1.1 Services & Request/Reply

Stateless protocols are communications that do not persist any session state and deal with self-contained requests only. Prominent examples are IP and HTTP. As such, they lend themselves to implement services with a request/reply pattern where requests contain all necessary information and can be handled in isolation, potentially followed by a reply being sent back and thus concluding the transaction. Such services are categorized as synchronous communication as the sender of a request usually waits for a reply before continuing. This can but does not necessarily imply that the process of the sender is also halted until a reply is received.
When used to retrieve data such a service is comparable to polling, meaning any requesting party must send a new request if it deems an update is necessary. This can make communication inefficient as it prevents a dynamic and reactive exchange of information that is triggered only when actual changes happen.

4.1.2 Event-Based & Publish/Subscribe

Event-based or event-driven communication is a form of asynchronous communication, meaning event messages can be sent whenever they occur and the receiving end decides when to handle them. They can be emulated over stateless protocols, but in order to overcome the continuous polling problem they typically rely on a stateful communication channel that is kept open for continuous exchange - TCP and Websockets are common examples for protocols. A full-duplex protocol facilitates a simultaneous exchange of event data on both ends.

The publish/subscribe pattern is an event-based communication and data distribution pattern. It involves participants providing data by *publishing* it. Other participants can then announce interest in certain (patterns of) events by *subscribing* to them. This forms a 1-to-n distribution of events between publishers and subscribers.
Routing of these messages between participants can take different forms. If all participants form a decentralized peer-to-peer network, each would typically subscribe at the publisher itself and receive messages directly. Alternatively, a network around a centralized message broker receiving all published messages and distributing them to subscribers can be established. The second form is better suited for cases with very heterogeneous participants where establishing bi-directional peer-to-peer communication between them might be difficult. Both solutions of course slightly differ in their behaviour, performance and scalability and each topology has its benefits.
As early as 1987, a publish/subscribe-like communication pattern for distributed systems likened to a billboard has been described in [101]. Emphasis was put on its flexibility and robustness while being easy to use for developers of applications. Indeed the pub/sub pattern is very appropriate for decoupling data producers and consumers in distributed interactive systems as well as potentially scaling message delivery to many interested parties.

Eugster et al.[102] differentiate all pub/sub systems against other interaction schemes for their ability to decouple across space, time and synchronization - meaning in order to exchange messages participants do not have to know about each other (i.e. know their network addresses), do not have to be present at the same time and their execution or message processing is not dependant on processes on the other end of the connection.
They further discriminate pub/sub systems into topic-based, content-based and type-based. In topic-based systems the identifier used to categorize message events is called *topic* (typically in the form of a character string). It is also used during subscriptions to express interest in events under said topic. Topics do not guarantee exclusivity per se, meaning that depending on the implementation it is allowed for different participants to publish their events under the same topic as well as different forms or types of events being able to be published under

the same topic. In many cases however, it is beneficial to ascribe a topic to a single publisher ("owning" the topic) and keep the type of data being published consistent in order to avoid confusion or ambiguity.
For content-based systems, subscriptions are handled based on what information is contained inside any message event. Subscriptions then take the form of a condition on certain message data fields/properties - for example an arithmetic or logical comparison describing a condition, threshold or distance metric based on event values like price or location. With content-based approaches an efficient and scalable implementation of these condition checks becomes very important to the overall performance of the messaging system. [103][104]
In type-based systems, subscriptions are registered based on the general format or type of event sent. On a social platform one could for example express interest in all messages that describe and inform about other users or persons. This requires that a set of known message/event/data formats is shared between all parts of the system whereas in a topic-based system the broker doesn't necessarily have to know anything about the payload contained inside message events.

Another relevant issue for publish/subscribe message distribution is message persistence, i.e. a solution to circumstances where participants join "late" but are still interested in or even required to receive past events. These past messages often reflect some form of global state which is necessary for other participants and is not updated frequently because it involves large amounts of data and/or does not change over time, meaning it is enough to publish only once. We will see in the following examples how implementations solve this issue.
How participants know about or agree upon topics and what format or protocol the data being sent adheres to are questions addressed via different mechanisms.

As we've seen in this section, the pub-sub method of message dissemination seems very adequate for many of Ubiquitous Mixed Reality's use-cases. It reflects the decoupled, dynamic and distributed nature. Some activities of Ubiquitous Mixed Reality need to happen for all parties involved and independently of them, so offloading certain computations into edge-computing processes or separate nodes that exist outside of individuals is a must.
Building ad-hoc networks - while being an interesting prospect for some cases - would require a lot of data replication on all participants (even when they individually don't need it) to guarantee availability of data in case one or more participants leave or drop connection.

The reason why Ubiquitous Mixed Reality and pub/sub are a good fit are as follows:

- In an environment that is dynamic, where entities come and leave as they please and its not clear who is present at a certain point in time, we need spatial decoupling. It is also not clear when and if somebody is currently

present, so temporal decoupling is important in certain circumstances. To assume we can interfere or have influence over execution flow of external systems in such an environment is also out of the question - i.e. synchronization decoupling is a must.

- In an environment that is not a-priori known to participants, they need other ways of identifying and expressing interest in certain aspects of the environment and respectively the relevant data channels. Pub/sub systems can provide this via topic/type/content subscriptions.

Naturally, event-based communication also has its weaknesses, one of them being large amounts of continuous data where the event-wrapper of individual event messages generates an accumulated overhead. This leads to the next communication pattern to be considered.

4.1.3 Dedicated Streaming Protocols

With event-based messaging, there is typically a bit of overhead for each event, for example additional information about the event's origin/identifier or what type of events are contained inside the message. There's possibly additional overhead (un-)packing messages before/after accessing event data inside. In cases where large amounts of data are sent with high frequency, this overhead can become costly. Sometimes data also occurs in a continuous flow with interdependant time sequences instead of asynchronous events. Video streaming is a prime example for this category of communication.

Such streaming protocols are typically very specific and geared towards special data/application cases in order to achieve better compression rates or latency. They are thus not prominent when considering the design of a dynamic and generally applicable base communication of a framework but are an essential consideration when it comes to how such a framework can support their use-cases.

4.2 Publish/Subscribe Systems

4.2.1 Apache Kafka

Apache Kafka [105] is a distributed messaging system originally developed for LinkedIn with a focus on log data stored on hard drives.

For log events (e.g. user activity on a website) it is required for consumers to be able to do data analytics possibly days after events occur. Therefore the entire event history needs to be saved persistently to the filesystem to be

processed at a later date. This is why Kafka prefers data consumers to pull messages over them receiving push updates. That way consumers can throttle message bandwidth to their processing speed.
In comparison, any MR environment is more transitional, immediate and interactive. Usually there is less use for analysis over event histories (e.g. how many times an element was interacted with) and only the last update or the current event itself matters (e.g. 6DoF pose updates, interactions with surroundings). Event history only becomes interesting for MR when thinking about testing or training (recording and replaying sequences), predictions / extrapolations or "time-travel" scenarios. One should definitely think about how to accommodate these use-cases but they can not be considered the default operating mode.
Nevertheless, some of the architecture and performance decisions of Kafka can inform decisions on MR systems. Kafka puts more emphasis on throughput rates and scalability than quality-of-service guarantees. These priorities are certainly adequate for MR as well for the above reasons - for example the update frequency on pose data is naturally very high and missed updates are rather inconsequential.

In Kafka, log events are topic-based. It is also inherently distributed, featuring multiple brokers that split topics into partitions for efficient load balancing of consumers requesting data. Kafka also handles batches of individual log messages as one transmission which limits message overhead.

For real-time stream processing demands, Kafka features options to apply time-windows to messages and join different streams of messages. In MR scenarios, mechanisms like these are relevant for interpolation/extrapolation algorithms (time-windowing) as well as establishing interactivity depending on separate streams of data (joining) - i.e. events of differing types coming from varying users, clients, processes, etc. that need to be observed in combination with each other because they should form relations triggering certain reactions. Especially the dynamic joining of individual data streams is interesting for delivering a combined package of all the necessary base as well as context information to processes doing advanced reasoning and interactivity in MR.

One benefit of persistent filesystem storage for events is that brokers can potentially crash, restart and still have old data present. In many MR cases a broker crash seems rather inconsequential as clients reconnecting after a broker crash would just continue to push their latest events (e.g. position) but as soon as keeping context between isolated use-sessions of a MR app becomes an issue persistent storage needs to be addressed. An effective and well-performing link between any event data and storage solutions (Databases etc.) is therefore relevant to MR as well.

Kafka uses Avro (4.4.2) as a data serialization system for delivering messages in binary format.

4.2.2 ROS

The Robot Operating System[106][107] (ROS) is a widely used open-source standard for robotics applications in industry and research. It is therefore quite relevant to understand how ROS solves connecting the sometimes heterogeneous computational components of (groups of) robots and how it manages and distributes the processing elements required to achieve necessary tasks.

To establish communication, ROS builds a graph of nodes instantiated on each of the individual components. Each node can be regarded as a processing unit with a distinct purpose. All nodes initially connect to a master node that organizes and orchestrates them. After recognizing other connected nodes all further data exchange between nodes happens through direct peer-to-peer connections negotiated through the master node. Data is distributed between the nodes via a topic-based publish-subscribe mechanism.
A special case exists for elements that can not open their own networking sockets for others to connect to via peer-to-peer because they lack the technical means to do so - browser applications are a good example - but nevertheless want to communicate with the ROS infrastructure. These cases can rely on rosbridge and its websocket server as a means to still exchange messages via publish-subscribe mechanisms.

Data formats are defined in special message schema files which are then compiled for all languages used by the different nodes. These schema definitions build upon primitive data types as well as previously established schemas which makes them continuously extendable. Users are then able to adopt and to build upon already established schemas.

To solve the issue of nodes receiving data being published in the past, ROS allows topics to be flagged as what it calls "latched". This results in the last event published being stored and any new subscribers automatically receiving the last event upon subscription.

ROS can fulfill quite a few of the requirements from chapter 1. It is however lacking in some other aspects. For example, for establishing communication typically the setup and number of interacting elements in the environment of the robot is assumed to be known.

4.2.3 DDS

The Object Management Group[26] (OMG) provides an open standard for the Data Distribution Service for Real-Time Systems[27] (DDS). DDS is a pub-

[26] `https://www.omg.org/` (last accessed 26-03-2022)
[27] `https://www.dds-foundation.org/` (last accessed 26-03-2022)

lish/subscribe middleware for distributed systems with industrial IoT in mind. It is designed for low-latency, reliability and scalability. [108][109]

4.2.4 MQTT

MQTT is a network protocol standardized for the IoT. It too builds upon the publish/subscribe pattern with a centralized broker. There is a list of available MQTT broker implementations[28] with varying software distribution models and sets of features. As a general solution for IoT the MQTT broker is typically topic-based.

4.3 Data Processing in Event-Driven Architectures

Ubiquitous and distributed systems often involve headless devices (IoT), devices with limited hardware resources (mobile or robotics applications) or the simple requirement to establish system logic and functionality governing certain parts of the system or providing services in a neutral fashion independent of any single directly involved participant. They form a network of (sub-)systems with data flowing between them via agreed-upon channels. This decoupling of functionality also has the advantage that it can potentially act as a library of reusable and combinable blocks with clear interfaces and modularity. On the other hand easy-to-use and adequate mechanisms to link and integrate these blocks with the rest of the participants is required. The more dynamic the structure of the overall system becomes, the more complex it becomes to establish these links and have them adjust to changes.
The following will take a closer look at some of the systems from the different fields that successfully implemented such mechanisms. They follow paradigms such as reactive programming, flow-based programming ...

4.3.1 ROS

The Robot Operating System, too, offers ways of distributing and sharing computational resources.
The example described by Quigley et al. in [106] (II.A.) presents a scenario of industrial robots that have multiple on-board computers networked together but also need to off-load computationally intensive tasks to more powerful hardware not located on the robot. This example illustrates it is sometimes necessary to offload computation onto different hardware. For these cases of distributed

[28] List of MQTT brokers: `https://mqtt.org/software/#servers-brokers` (last accessed 26-03-2022)

processing and control, ROS nodes can provide functionality to other nodes. Interesting to note are the different modi in which computation can be triggered (also see [107]).
The first case is to simply establish a node subscribing to certain topics, processing data, then publishing the results - this is the fully asynchronous mode in which the node continuously does its work without additional outside influence or control.
As a second alternative, nodes can provide fully synchronous **Services** where one node would post a request and await the result.
In cases of long-lasting tasks that should not be blocking execution for other nodes and/or require progress & status updates during their execution, ROS provides the concept of **Actions** that can be requested and will eventually provide a result.

4.3.2 Node-RED

Node-RED is built on Node.js and is designed to connect together devices, APIs and online services. It allows users to combine functional blocks called nodes into data flows to handle and manipulate event data between these endpoints. Nodes representing inputs, processing and outputs can either be taken from an existing library or new ones can be written in Javascript, then potentially published for others to import into their library. Flows can be created, changed and deployed in a browser-based editor. In the same editor, individual nodes belonging to the flow can be configured or edited. The editor follows a visual programming approach. Node-RED can be deployed on different scales from small personal devices to cloud servers.

4.3.3 Neurorobotics Platform

The NRP has two components that are essentially event-driven. On one hand, there is the robotics simulation based on ROS and Gazebo (until v3.2, more engines from v4.0 on), both of which communicate their events via pub/sub messaging. On the other hand, there is the brain simulation running on Spiking Neural Network soft- (NEST[92], Nengo[90]) or hardware (SpiNNaker[110]). Neuromorphic architectures - whether in hard- or software - are event-driven in nature (spikes) and the event-driven processing is a core feature.
To combine both (originally) indepentent systems, the NRP uses the concept of Transfer Functions (TFs) forming an event-loop between both ends that is continuously evaluated and keeps both systems and their signals/events synchronized. TFs are pieces of code that can translate sensors data (e.g. camera images) into spikes and vice-versa (e.g. spikes into motor control).
As such, TFs act as translator modules mapping events from one domain to the other.

4.3.4 UbiTrack

UbiTrack[111] is a framework tackling the challenge of tracking in ubiquitous environments containing heterogeneous sensors.

It determines optimal data flows in a peer-to-peer network through Spatial Relationship Graphs (SRGs) - nodes representing local coordinate frames and edges specifying spatial relations. Notably, implicit spatial relationships can also be deduced.

Trackman[112] is a GUI tool for UbiTrack that provides a library of patterns and modules to apply to tracking tasks. Visualizations of SRGs help with the identification and setup of computational patterns.

4.4 Message Descriptions and Formats

As argued before, MR/ME is and will likely stay a heterogeneous, multi-standard and multi-purpose conglomerate. It follows that systems or applications existing in an MR context will be following different data formats that offer the right level of abstraction level and fitting performance for their purpose. So, apart from the question of how best to deliver messages between the parts of a mixed reality system, a common basis for communication needs to be found.

In Section 2.2.1 and 2.2.2 virtual and mixed environments were characterized in their different levels of complexity. Agent-based models were also quoted with the need for an agent-communication language. With extremely static VEs like videos or even just 3D models that do not change over time, any environment can be constructed and mixed with this static format as long as an importer/translator for the file format exists. Analogously, for a dynamic, event-based and reactive mixing of environments a common language is required to describe these events and enable every environment to interpret them.

These data format protocols constitute the common vocabulary between heterogeneous systems. And here two problems form on both ends of communication: a) finding an efficient internal standard understood by all parts of one system/application developed by one team for one purpose and b) finding efficient ways of communication with external systems and standards developed by other teams. Besides an internally consistent way of communication when building MR applications, it is therefore also prudent to think about how this application will be able to communicate with external standards and formats - essentially how foreign vocabularies or "dialects" can be learned and interpreted or how translators between both can be found and established.

The structure in which a message is being transmitted will be called message

format. The way of describing such a format to an interpreter trying to read or formulate valid messages will be called message schema.

4.4.1 JSON

JSON[29] (JavaScript Object Notation) is a data interchange format based on human-readable text defining name-value pairs. It is language independent and supported by all modern languages. Being interchanged between systems in its text form, it is usually not the most bandwidth-efficient solution for data exchange. To reduce space requirements, solutions like BSON[30] (Binary JSON) exist.
While it offers syntax and primitives like objects, arrays, strings, numbers, booleans and null, higher level semantics and schemas of data type definitions beyond the standard sets that are required for (de-)serialization or validation of messages during exchange between systems require additional setup like JSON Schema[31].

4.4.2 Apache Avro

Avro[32] is a message schema description and binary (de-)serialization system. Kafka integrates it for its message formatting purposes. Its message schemas are written as JSON and are evolvable. It can also act as a container format when writing files.
One distinction of Avro is that the schema describing the data format is always transmitted or saved together with data and thus schema compilation into code for different languages is not mandatory but can be done for optimization purposes.

4.4.3 Google Protocol Buffers

Google's Protocol Buffers[33] is another way of establishing message formats via schema description files which are then compiled to all relevant programming languages. Message objects are (de-)serialized into binary before and after transmission respectively.
A performance evaluation comparing Protocol Buffers to JSON/BSON for IoT is available in [113].

[29] `https://www.json.org` (last accessed 26-03-2022)
[30] `https://bsonspec.org/` (last accessed 26-03-2022)
[31] `http://json-schema.org/` (last accessed 26-03-2022)
[32] `https://avro.apache.org/` (last accessed 26-03-2022)
[33] `https://developers.google.com/protocol-buffers/` (last accessed 26-03-2022)

4.4.4 YANG

YANG[114][115] (Yet Another Next Generation) is a data modeling language reflecting its data models (or schemas) in so-called modules. Modules can include and augment other modules as submodules and track revision history, thus any defined model can be extended.

Modules can be hosted as a web resource, their URL serving as their reference for others to import. This allows any system to use and understand publicly available modules.
Also worth noting, multiple parties can theoretically implement the same model under different URLs without producing conflicts. Standardization of such an overlapping model at a later point in time is possible without much additional effort.

4.4.5 Zero-Copy Formats

Both Avro and Protocol Buffers presented above involve a (de-)serialization step converting from programming language specific message objects to binary before sending and back from binary to language object after receiving a message. This of course means additional processing overhead for each message. Solutions like Google's Flatbuffers[34] and Cap'n Proto[35] eliminate this (de-)serialization step by aligning message fields in memory and addressing individual fields of a message via fixed (e.g. 32 bit floats) or pre-calculated offsets (e.g. strings). The goal is to gain increased performance by omitting these steps. In short, a message is already written in a serialized fashion while assigning values to its fields during building/definition of a message object. Conversely, after receiving a message individual fields can be accessed directly through their offsets without prior deserialization. Compiled message objects per language now essentially are reduced to getter/setter functions storing the respective offset for a field and reading/writing data to the appropriate places in memory.
Some performance comparison and additional investigation on hardware acceleration has been done in [116].

This, however, also means that message construction and modification is much more rigid and complex. If, for example, messages should be treated as communication of states with intermediate processing steps changing parts or recombining fields of separate messages into new messages, then handling via these methods becomes much more difficult. Messages in their memory-aligned binary format usually have to be reconstructed or recombined from scratch to guarantee proper alignment. Together with Leon Sandner in his thesis "Ubi-Interact:

[34] `https://google.github.io/flatbuffers/` (last accessed 26-03-2022)
[35] `https://capnproto.org/` (last accessed 26-03-2022)

Architecture for Programming Language Independent Interaction Modules"[36] performance tests were conducted comparing Flatbuffers to Protocol Buffers. An overall performance advantage in latency of about 10-15% was observed. It became also clear, however, that for certain languages memory access/copy was implemented using sub-optimal methods which was especially noticeable for larger byte data like images, resulting in a net performance loss. These issues are surely fixable over time. What is a more problematic downside is the natural complexity for handling messages dynamically. This essentially resulted in helper and utility functions being implemented to save time and effort during coding and make code more readable and manageable, negating some or all of the original performance gain.

4.5 Agents, IVAs, AI

The virtual world can be populated with simple entities whose properties are presented to us and which we can manipulate to some effect through simple interactions. Eventually though, agents should enter the environment(s) that can perform more complex and helpful tasks and assistance. Such agents are usually referred to as *Intelligent Virtual Agents/Assistants* (IVA).

A definition of an agent can be adopted from Wooldridge and Jennings [117]. Specifically, the weak notion of agents with properties of autonomy, social ability, reactivity and pro-activeness seems sufficient for considerations about how they could navigate Ubiquitous Mixed Reality scenarios. Human-like characteristics might be helpful in some use-cases but are not generally required here. Russell and Norvig [118] also characterize agents by their ability to perceive their environment and act upon it through sensors and effectors.

Norouzi et al.[119] investigated the convergence of the fields of augmented reality, the internet of things and intelligent virtual agents, describing their combined potential as transformational. They note how with increased contextual awareness the usefulness and range of possible tasks of an IVA also increases. IoT is identified as an opportunity for IVAs to observe and influence the physical world and to pervasively integrate - also noting that this requires new mechanisms for privacy, access control and sharing of awareness, appearance and abilities. AR represents the possibility to seamlessly blend or embody an agent into the user's surroundings but also for a user to utilize contextual and natural forms of communication, building trust and understanding about which tasks the agent is currently performing.

Chung et al.[120] specifically look at one of the more wide-spread consumer

[36] Sandner, L.A. (2020). *Ubi-Interact: Architecture for Programming Language Independent Interaction Modules* [Unpublished Bachelor's Thesis]. TUM. `https://wiki.tum.de/display/infar/%5B19WS+-+BA%5D+Ubi-Interact%3A+Architecture+for+Programming+Language+Independent+Interaction+Modules`

devices today with regards to privacy concerns for cloud-hosted services. They demonstrate how additional contextual information and profiles about users like time of use or behavioural patterns, personal interests and preferences and travelling routes can be extracted with continued use of the system. A discussion about user security&privacy risks that go beyond simple data security&privacy is added with potential measures against it, such as regularly triggering wipes of usage history and logs from the cloud servers.

4.6 Digital Twins

The subject of digital twins are a related field in which the mixing of environments becomes very apparent and a single physical object or device becomes heavily intertwined with its digital counterpart. Instead of the term *physical/virtual environment*, these concepts are also referred to as *physical/virtual space*, *cyber space*, *Cyber-Physical System (CPS)* and *Cyber-Physical Production System (CPPS)*. [121][122][123][124]
Furthermore, their concept mostly occurs in the context of industrial manufacturing and optimization and is applicable to many/all stages in the life-cycle of an object [125][126][123][122][124]. The individual objects are thus embedded in a bigger environment of communication, analysis, prediction and control.
Concepts and hardware typical to VR, AR or MR research can play a role for visualization and interaction but are not necessarily the focus. The research field of digital twins serves as another example where a clear differentiation between the (objective) environment and the (subjective) reality is made.

For a technical foundation to realize digital twin systems, Tao et al.[121][122] identify five dimensions that are important: 1) the physical entity, 2) the virtual entity, 3) a shared data buffer at the center, 4) services with data access that both the physical and virtual entity can rely on and 5) a connection part bridging to all elements.
A mixed reality & environments framework certainly reflects these dimensions in some form if it wants to achieve a tight integration of environments.

4.7 Context-Awareness

This section will present some more detail and solutions to acquiring contextual information and ways to use it. This is not because of the relevance of the individual solutions for the rest of this work but more to give an overview and discussion over what mechanisms a framework needs to provide and integrate with to be viable for the task of context-awareness for UMR.

Figure 10 gives a rather humorous example of a visual scene where context helps making sense. Machine Learning in particular, though, illustrates that

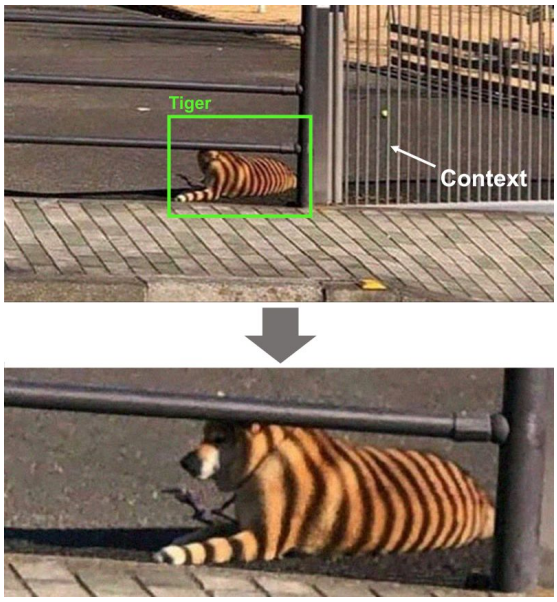


Figure 10: Visual example for context - LinkedIn post by Ralph Aboujaoude Diaz

context is an integral part of what is being learned and understood. [127]

Context in its various definitions [37][38][39][40] can be interpreted as the environment or circumstances under which an event happens that help characterize the event and its relations with and effects on the environment. It helps not only to explain events but is often enough the only way to achieve proper meaning and avoid misinterpretation. The relevance it plays for making proper observations and useful conclusions in any reality is therefore obvious.
Sato defines it as follows: "Context is a pattern of behaviour or relations among variables that are outside of the subjects of design manipulation and potentially affect user behaviour and system performance". [128]
Bricon-Souf and Newman[33] refer to several other definitions and give an analysis framework for healthcare applications. They identify three main axes:
1) The purpose of use, meaning context can be used to a) present information and services to a user, b) execute a service and c) tag information for later retrieval.
2) The items of representation, i.e. mainly people, environment and activities.
3) The organization of features, i.e. are features of context following a) hi-

[37] `https://www.merriam-webster.com/dictionary/context` (last accessed 26-03-2022)
[38] `https://www.thefreedictionary.com/context` (last accessed 26-03-2022)
[39] `https://www.dictionary.com/browse/context` (last accessed 26-03-2022)
[40] `https://dictionary.cambridge.org/dictionary/english/context` (last accessed 26-03-2022)

erarchical ordering from general to specific, b) conceptual ordering of internal aspects (mood, etc.) vs. external aspects (temperature, etc.), c) ordering based on the focus of current activities or d) a usefulness ordering based on relevance for current actions.
As described in Section 2.2.3, in general context information can come from ambient sensors and environment models as well as individual devices/systems/users.

For UMR applications it is then imperative to be context-aware, especially if they are supposed to operate in everyday life scenarios. If realities are supposed to mix, they need to be able to relate themselves to the context of the other reality. Especially the capabilities, affordances and internal context/state of virtual entities - even if they have a physical component to them - can be quite hard to decipher for humans and other systems if the entity does not give concise clues and feedback about its processes, state and how to approach it ([40], Thesis 38+39).
To make these things more apparent and relatable, a framework needs to provide standardized ways to describe and connect to such interfaces and give form to the virtual aspects.

Apart from the typical live context an application is running in - e.g. location, individual and social circumstances or available devices - debugging can be seen as an extreme example that (hopefully) does not belong to everyday use-cases but demonstrates the difficulty one might encounter trying to understand virtual processes. Debugging in itself, whether from the perspective of a simple user or a developer, is a complete switch of context by which the application is approached and usually requires far greater insight into what is happening and why, especially in mobile distributed applications ([40], Thesis 41+44+46).
For a framework to be viable long-term and with growing complexity, it needs to provide accessible and powerful tools of introspection to developers and appropriate warning and error communication channels to participants. It furthermore needs to be aware of its own operating context to be able to detect potential failures early or give indicators and logs about sources of errors.

For improved 3D scene understanding with localization and reconstruction, the works of Bowman et al.[129], Dai et al.[130], Fehr et al.[131] and Tahara et al.[132] show increased focus and consideration for integrating visual semantic information. These solutions typically rely on a mobile system like an HMD or a robot equipped with cameras. They generate and integrate the semantic information together with the geometric reconstruction from SLAM-based[133][134][135] approaches as this integration helps both sides reach better solutions.

For the system of *RagRug* providing a framework for situated analytics in AR, Fleck et al.[136] also utilize reactive programming mechanisms to provide necessary real-world context for AR visualizations and user interaction. Targets for context-awareness are movement of physical objects to adjust placement etc.

of the augmented visualizations, changing light conditions to adjust display and rendering conditions, inferred targets for user commands improving ease of use for the system and error detection to warn about invalid actions.

For any environment, some participants may not be equipped with all necessary capabilities to generate certain contextual information. A framework or overarching system that aims to integrate and provide visual (semantic and geometrical) information - and for that matter any other context - thus needs to provide pathways to retrieve, store and consolidate such information from the capable elements like mobile cameras.
Mobile devices, though, may not be present at all times and/or in all desired directions while stationary/ambient devices can not rely on dynamic exploration and mapping. In order to provide a more complete picture, a UMR system will probably rely on the continuous effort of multiple devices over time accumulated in an instance independent of each individual element.

Intelligent Virtual Assistants/Agents (IVAs), too, greatly benefit from contextual enrichment and integration. Indeed, many cases of assistance they are designed and envisioned for improved personal relatability and engagement or providing helpful information relevant to an open-ended situation are entirely impossible without proper understanding of the situation they are operating in. [137][138]

Norouzi et al.[119] describe how the fields of Augmented Reality, the Internet of Things and Intelligent Virtual Agents can benefit each other in the future. They describe a possible outcome of such a convergence as an Augmented Reality Agent (ARA) with awareness of the surrounding physical world and potential to influence it through IoT devices, seamlessly blending in. This again illustrates that such systems - while being physically, logically and/or computationally separate because of the sheer complexity of each system alone - can not exist in isolation.

Activity Recognition is one recognized problem for ubiquitous computing applications, relying on e.g. accelerometer and GPS data from smartphones. [139][36][35]
For ubiquitous computing in general, according to Schilit et al.[32] context is also quite essential to navigate and discover the environment and help mediate interactions. They investigate mobile distributed computing systems which they describe as a collection of mobile and stationary computing devices that are communicating and cooperating on the user's behalf with the aim to provide ubiquitous access to information, communication, and computation. Context gives information about which elements are of interest or important and should come into the focus of attention at a given time, which devices are fit to help reach a certain goal, where the user is and who he is with, and so on. Other important contextual information they list are "... lighting, noise level, network connectivity, communication costs, communication bandwidth and even

	manual	automatic
information	proximate selection & contextual information	automatic contextual reconfiguration
command	contextual commands	context-triggered actions

Table 1: Context-Aware Software Dimensions as described by Schilit et al.[32]

the social situation" ([32], p.1). As Schilit et al. also note, this ubiquitous computational environment is constantly changing. It should therefore be vital to systems embedded in a Ubiquitous Mixed Reality world to be able to inform about their purpose, their profile, their interface, etc. as to be identified, found and used properly.

In remote collaborative work scenarios where virtual body posture play a big role, the distance or general visibility of other participants could indicate which body pose data updates are actually required by whom and which updates can be currently omitted to save bandwidth. A network analysis of a current state-of-the-art collaborative workspace system by Ruizhi et al.[140] shows possible shortcomings in terms of scalability.
In order to make sensible use of given resources like bandwidth and computational power in mobile and ubiquitous scenarios, the depth and type of information that is being transferred between systems should be adjusted according to contextual factors.

To describe, build and categorize context-aware applications, they use two orthogonal dimensions. They differentiate by 1) whether a task is retrieving information or execution commands and 2) whether a task is executed manually (user input) or automatically. Multiple challenges are identified with each category according to Table 1. For *Proximate Selection* combining different dimensions of contextually relevant information, e.g. location and order, together and finding an appropriate UI display method can be difficult, especially with limited screenspace on mobile devices. With *Automatic Contextual Reconfiguration* rapid changes in context might negatively affect system performance as well as confuse or distract users so it may be necessary to narrow down relevant context based on the situation and/or user's task and intention. Especially with *Contextual Commands* it was commented that service providers like businesses and government providing such a command interface could extract contextual information, thus gaining access to or being able to generate personal user profiles unintended and unwanted by the user and using/abusing them for personalized advertisement or assistance. The main issues with *Context-Triggered Actions* are identified as balancing timely execution versus predictable behaviour as well as the expressiveness of a predicate language in combination with the accuracy

and timeliness of underlying context information.

As vitally important as context-awareness is, the thought of a large-scale Ubiquitous Mixed Reality system where a centralized authority monitors, provides and possibly even generates context can quickly lead to scenarios with massive security, privacy and ethical concerns. It might not even be feasible from a technical side to establish such a singular "big brother" that is capable of accumulating the full context, as some contextual information may be intrinsic to systems and communicated to the outside world only partly or not at all. Some of the context might be more abstract, something that for example an image processing algorithm can't extract on its own (location gives hints that a tiger in the middle of the street is very unlikely, agent formulates what their intention/goal is at the moment).
If we instead assume that a Ubiquitous Mixed Reality world consists of several individual systems, context should rather be something that is communicated, formed and emergent between systems. This again reinforces the importance of Ubiquitous Mixed Reality systems being able to find adequate common communication grounds with foreign systems - much like interpersonal communication. This can be done through standardized data exchange formats or tight integration of available interfaces to the "outside world" as to be able to integrate external events into the internal feedback loop. If a Ubiquitous Mixed Reality system is not built to do this effectively, it will stay isolated and "lonely". Standardized formats naturally form over time and plugins for external systems are developed for each system as necessary - the important part is the internals of the system being engineered to work with external sources effectively and flexibly.

4.8 Security & Privacy

Already today, when and how big internet service providers or government agencies make use of personal data is a big topic of discussion. If technology and services are integrated in a ubiquitous way intertwining with physical reality, these problems even take on new qualities and can potentially become much more intrusive and concerning. Especially when it comes to medical data, the topics of confidentiality and consent become prominent ([141]).

Figure 11 gives a general overview of security and privacy properties and their threats pertaining to MR scenarios. Not all properties can be guaranteed at the same time as some security demands conflict with privacy considerations - as illustrated by non-repudiation or anonymity being a possible property and threat simultaneously. This means one has to find an acceptable trade-off between security and privacy in some cases. [142]
Gabriel et al.[143], too, note that "[t]here is an inherent conflict between pervasive computing's goal of accurately identifying persons, objects and messages

	Property	Threat
Security-oriented →	Integrity	Tampering
	Non-repudiation	Repudiation
	Availability	Denial of Service
	Authorization	Elevation of Privilege
	Authentication	Spoofing
	Identification	Anonymity
	Confidentiality	Disclosure of Information
← Privacy-oriented	Anonymity & Pseudonymity	Identifiability
	Unlinkability	Linkability
	Unobservability & Undetectability	Detectability
	Plausible Deniability	Non-repudiation
	Content Awareness	Unawareness
	Policy & Consent Compliance	Non-compliance

Figure 11: Overview of general security and privacy properties and the respective threats to them as presented in [142].

(authenticity), and the desire for anonymity – to prevent data trails from the outset".
Bellotti and Sellen[144] "[...] take privacy to be a personal notion shaped by culturally determined expectations and perceptions about one's environment" in light of evolving social norms and practices determining what is acceptable and at what point benefits outweigh risks.

Many of today's security and privacy concerns are focused on the world wide web as the major technology for open and widely available international communication. Internet technology offers several protocols, layers and mechanisms as solutions to these risks. Yet they are commonly focused on one-to-one client-server communication. For any web-hosted service, securing the web server, the end user's client device as well as the communication between them has to be considered. [145]
To bolster the integrity of communication channels, they can be encrypted via SSL/TLS and HTTPS or SSH.
For the issue of authentication of users/processes/devices and their consequent authorization of resource/data access, one can rely on open solutions like OAuth ([146]), OpenID Connect (OIDC, [147]), Security Assertion Markup Language (SAML, [148]) , JSON Web Tokens (JWT, [149]) and Keycloak ([150][151]).

The field of IoT also exemplifies several additional considerations. [41][152] In general, the heterogeneity and increased scaled in terms of numbers of devices increase security threats. As a complex system of interconnected devices any

disturbances carry a higher chance for unpredictable cascading effects. Confidentiality is harder to fulfill when messages can take several stops in a chain of nodes and processes where originator and authorization are not checked in every step. Any communication of devices without identity and permission checks or without notifications to owners can be problematic. Conversely, the exaggerated time, performance and energy constraints on small-scale or embedded devices make security checks using traditional methods harder. Especially when working on streams of large amounts of data, excessive overhead must be prevented and light-weight algorithms are preferred.
The dynamic and often ad-hoc nature of their environments also requires new methods of establishing trust between participants when there's no central and certified entity. Unexpected environments also require more adaptation ans self-healing capabilities in order to make networks more resilient. Information that might otherwise be confidential like location may change its confidentiality during an emergency situation when help is needed and the disclosed location might be invaluable.
IoT devices also hold the potential to pervasively surveil and identify users without their control, consent or knowledge. Together with profiling over long periods of time it is another threat to privacy, the mere prospect of which is often enough to produce aversion in users towards such technology. The presence of a certain set of devices alone can be enough to identify a place or person if these devices are openly addressable (inventory attack). Another cause of unwarranted disclosure may be the change of ownership or shared use of devices. Linkability is also at a higher risk in a distributed and pervasive environment that integrates different sources of data over time.

For social aspects, Bellotti and Sellen[144] comment that besides insidious exploitation, the issues of disembodiment and dissociation received relatively little attention while potentially being much more pervasive as they are unintentional.
Disembodiment occurs when there is no technical means of communicating additional contextual information like body posture, facial expressions, gaze direction or voice level/intonation. It also refers to uncertainty and a lack of feedback about when and to whom information is conveyed. Dissociation means the that only results of actions and not the actions themselves are presented so that the originator or cause can not be determined.
Another consequence of new technology may be the breakdown of established social and behavioural norms by violating principles like "if i can't see you, you can't see me" when there is no control over who can observe one's actions at any given time. This loss of personal privacy is also mirrored when others inadvertently intrude private space due to not being sure about one's availability or willingness to interact.
[144] proposes a design framework for applications that considers feedback and control in categories of information capture, construction (processing, storage), accessibility and purpose (might require inference).

Ubiquitous Mixed Reality with its pervasiveness and increased need of sensing shares a lot of the previous issues. Latent and bystander privacy is as much of a problem if a user's physical environment is constantly scanned, mapped and contextualized. A new quality is the focus on physical input, output and interaction with virtual content that may give away information or allow adversarial attacks. To name a few examples, a recording device may capture a display containing sensitive content from another application (passive/latent input) or gestures (active input), content may be perturbed (sometimes in humanly unrecognizable ways) to fool algorithms in their analysis [153] and malicious placement (e.g. obscuring physical dangers) or properties (e.g. flashing lights) of visually rendered content may affect users. For this reason [142] categorizes threats and protection mechanisms for MR into input, data access, output, interaction and device.

Besides many detailed technical solutions so these issues, there are some general concepts presented by the previous works.
A simplification of services and a separation of concerns draws better borders and limits the scope of data use. Sanitization layers can be introduced that clear e.g. latent and private content and data before passing it further down the pipeline. These sanitization layers can work with intrinsic parameters defined by the current application itself but also rely on extrinsic or context-based parameters defined and provided by the environment. Furthermore, abstraction layers can e.g. pre-process and classify events instead of giving direct access to raw data. A general policy of minimum disclosure by only requiring and communicating what is essential to the provided service for the minimal duration it is lasting is a good idea as well. In changing environments, Attribute-based access control (ABAC, [154][155][156]) can be utilized in this regard, evaluating attributes and characteristics of users, objects, environment(s), connectino and administration rather than assigned roles. If content is only to be provided with the express agreement or presence of all affected parties, secret sharing or secure multi-party computation may be an option.

In his work on dataveillance, Clarke[157] identifies major privacy issues whenever the possibility arises to cross-identify people and their saved data between platforms and institutions.
For a ubiquitous mixed reality, ways of isolating the use of private data is thus a must.

Recent developments like the Fediverse[158] based on federated and self-hosted hard- and software as a counterpoint to big centralized social media platforms are an expression such concerns.

4.9 Libraries, Tools & Platforms

4.9.1 ZeroMQ

ZeroMQ[41] (also ØMQ, 0MQ or zmq) is a networking library that can extend to perform as a concurrency framework, allowing inter-process as well as in-process message transport. It supports multiple communication patterns over different protocols (TCP, UDP, IPC, TIPC, multicast, WebSocket) like the aforementioned request/reply and publish/subscribe among others.

4.9.2 Node.js

Node.js[42] is a JavaScript (JS) runtime based on the V8 engine[43]. It is asynchronous and event-driven in nature, designed for real-time network applications. It furthermore allows cross-platform development.
While JS code execution itself is single-threaded, the V8 engine features an event-loop where non-blocking methods and especially I/O calls like filesystem and network access provided through native C/C++ libraries like *libuv* can be handled asynchronously via callbacks, thus enabling concurrency. *Libuv*[44], in turn, makes use of a thread pool for execution of tasks. This makes Node.js rather easy to use while still being relatively performant when applied to I/O and networking tasks. It is however comparatively weak for CPU-intensive operations.
Parallelism can be achieved through forking child processes or using worker threads, with data exchange between processes/threads being supported in a straight-forward - again event-based - fashion. This is further supported by JS libraries like *workerpool* that perform analogously to thread pools, supporting both Node.js and browser environments.

[41] `https://zeromq.org/` (last accessed 26-03-2022)
[42] `https://nodejs.org/` (last accessed 26-03-2022)
[43] `https://v8.dev/` (last accessed 26-03-2022)
[44] `https://github.com/libuv/libuv` (last accessed 26-03-2022)

5 Ubi-Interact

Following considerations about the structure and important issues of Ubiquitous Mixed Reality from the previous chapter, Ubi-Interact is an effort to connect environments. This of course means signals captured from physical entities to be related to virtual environments and vice-versa, but also connections between otherwise separate VEs as there may exist several - reflecting the heterogeneity and plurality of Ubiquitous Mixed Reality.
The goal is to build a framework capable of representing relevant base elements of environments in a useful and flexible way and then enable each environment to react to signals from other ones. Adapting to future developments, offering easy integration and fast results that can be incrementally built upon are the main goals relating to the use of the framework by developers.

The core focus of Ubi-Interact is on the communication between different systems and devices. Their individual strengths and capabilities should be leveraged in combination with each other. This forms them into a bigger distributed system that opens up new forms of interaction, benefiting from their mixed capabilities. At the same time, the individual parts and components delivering this data should be represented in a way that makes it possible to dynamically identify and adapt to changing circumstances as well as have individual parts be replaceable and interchangeable with a given profile of their affordances.

In its base communication functionality, Ubi-Interact can be used like a messaging and service middleware. It starts to act like a framework when concepts of Devices, Components and Processing Modules are used to provide form and functionality to a more complex system.

Ubi-Interact tries to keep the scope limited to communication and representation of communication interfaces as far as it is required to find adequate paths of communication and combination for Ubiquitous Mixed Reality. When concepts are introduced, the goal is to make them generally applicable and extendable so they stay useful without imposing restrictive or overboarding effort on developers.
It should always serve to offer quick ways of describing the context of what a certain part of the distributed system is designed to do and with that offer other parts the ability to identify foreign contexts without imposing limitations on the individual systems. This means that a given device or functionality - rather than working within the mindset of Ubi-Interact - should find it easy to add Ubi-Interact into its own process and use it to connect to the rest of the Ubiquitous Mixed Reality environment.

Whenever Ubi-Interact finds a beneficial application, it is always seen as a small addition to the multiplicity of Ubiquitous Mixed Reality and therefore must itself stay flexible in its communication and integration with systems not based on Ubi-Interact. Integrating other systems via control and orchestration

over them may be desirable, but there is equal focus on being integrated by other systems and offering ways for them to control the necessary / allowable parts of Ubi-Interact.

All code is available on Github[45].
Terms in italic refer to a Ubi-Interact class or concept and can usually be found as a Protocol Buffers schema under `https://github.com/SandroWeber/ubii-msg-formats/blob/develop/src/proto`.

5.1 Requirement Analysis

In [96] a list of general requirements for Ubiquitous Mixed Reality frameworks has been argued for. The following will take a closer look at how Ubi-Interact specifically is trying to address each issue.

Plasticity, Adaptivity

Plasticity is seen as a system's capability to be utilized in different environments, operate on different scales, work with moment-to-moment changes in its surroundings but also work with changes in the underlying hardware. In short, it is a criterion for how malleable a system is.
Adaptivity is regarded as the capability to automatically adjust internal configurations without requiring specific user input or defined preferences.

One of the main assumptions about the nature and future of Ubiquitous Mixed Reality from Section 2.2 is an extremely dynamic environment, constantly shifting in participants, technology, standards and so forth. This is already a challenge for developers in such an environment within a single homogeneous framework, but becomes even more difficult when considering external systems operating in the same environment and how to interact with them.
Especially with multiple actors that are very similar in their characteristics but are otherwise only directly addressable via a special unique identifier which is often quite cryptic to guarantee its uniqueness, it becomes less and less about predefined hard-wired addresses or values shared and assumed to be known by every party and more about being able to identify and navigate surrounding entities and their profiles.

As an analogy, we as humans recognize other humans, their faces and their common structures yet recognize their individual differences without having to know them personally. The same is true for tools and objects, whose functionality can often be worked out with a bit of inspection, trial and error. Different types of roads or pathways, too, can be judged to be fit to drive on or not.
Something similar should also be possible in a more abstract fashion for virtual

[45] Meta-Repository: `https://github.com/SandroWeber/ubi-interact`, individual repositories found under user `https://github.com/SandroWeber`

entities and their properties during navigation of a virtual reality, even though they might not have an associated physical structure or visual geometry that make them easily recognizable to us.
The question is then how to model the characteristics of virtual entities in a flexible manner so that certain profiles or characteristics can be observed by other entities (and eventually humans after one or more additional mediated steps) and they can spot potential tools, partners, etc. for their needs and purposes.

The desirable end-state for this issue would probably be a shared standard between systems for describing affordances of virtual entities and how to interact with them. As such a standard is currently lacking, Ubi-Interact is trying to investigate some common and useful structures that might solve some of the issues in terms of individual systems dynamically adapting to other systems that share the same (virtual) space.
The objective is not to define such a standard but instead to explore some of the components that may be helpful in this regard. Ubi-Interact uses it as an internal standard only, external systems that are integrated with can be described in Ubi-Interact's terms through their respective node though.

In terms of virtual entities expressing their properties, affordances or events Ubi-Interact utilizes the concept of *Components* and *Devices*, which will be described in more technical details later. Suffice to say for now that components describe the individual affordances in an extendable way - i.e. new fields and descriptors can be added as deemed necessary - and devices are used to freely define groups of components that logically belong together and form a unit.

Important to note here is that the format in which components and devices can be described can be evolved and that devices do not put any conceptual restrictions on which components can be grouped. This helps avoid circumstances where common devices must be described in a certain but static way that might not be applicable anymore when slight variations occur, e.g. a prototypical interaction device is exploring a new combination of components, the next generation of hardware has to be integrated with a new descriptor because it gained/lost certain aspects relative to the preceding iteration or comparable situations.

Instead the new profile can be adjusted and other entities can judge for themselves whether it still fulfills their needs. Components and especially their constellations as devices can then be searched and filtered to find your way around the virtual environment and identify ways of interaction.

The decoupled event communication gained through a pub/sub broker also contributes to a general plasticity in the system. Any communication between nodes only relies on a shared message type for their respective calls to work. Theoretically any such data source or sink can be re-mapped during runtime.

Connectivity

In a Ubiquitous Mixed Reality world, individual entities need to be able to connect with each other without either knowing about the other side in advance. They potentially also need to communicate across time with either side being absent at the time of events occurring. The pub/sub scheme (4.1.2, [102]) identifies these properties as decoupling in space and time and is therefore an ideal candidate to achieve these requirements of connectivity. This is confirmed by the many systems with similar characteristics adopting the pub/sub scheme ([46], [136], [106], [159], [160]).

Ubiquitous Mixed Reality should also show characteristics of persistence, thus systems need to run over extended periods of time - ideally 24/7 with fast recovery after a breakdown. For Ubi-Interact this means that continuous deployment and testing is required and that general (reconnect) and special cases (data/state persistence) of recovery after breakdown need to be considered.

Spatial distribution of deployed and mobile elements are another issue. Latency and inconsistencies of data channels are important considerations here. Ubi-Interact must not only work in localized networks but also be able to communicate world-wide. The performance required for real-time latency interaction must constantly be evaluated and mechanisms allowing to the system and developers to detect and adjust to fluctuations in latency should be considered.

Scalability

Different use-cases like IVAs (Section 4.5) in a personal room-scale environment up to ArenaXR[46] with a city split up into realms for localized content make it clear the ubiquitous part of Ubiquitous Mixed Reality is covering many different scales. Consequently, Ubi-Interact should be able to operate on different scales itself as well as be able to connect and integrate within several spatial scales.

Within a single category of spatial scale, it is also important to consider that client-independent processing like edge/cloud computing solutions be scalable to a varying number of attendants. With different numbers of participants it is of course also important to treat communication bandwidths efficiently.

Security, Privacy

From Section 4.8 it can be concluded that concerns for privacy arise with centralized platforms where cross-identification of individuals can be achieved between multiple activities without the individual having any control over said data.

In terms of security in Ubiquitous Mixed Reality, encryption is a given. For the channels of physical reality that can't be secured digitally where data can

be observed, spied upon or stolen intentionally or unintentionally, an increased situational awareness and contextual understanding of physical reality by the Ubiquitous Mixed Reality system is desired. The same awareness may also help with decisions about when/if to occupy users with new information in situations where distractions (from physical reality as well as from other more important information) could cause harm to them and others.

Ubi-Interact is therefore trying to keep individualized setups on user-controlled hardware possible, establishing time-limited context and sessions between parties that give their consent and trust with the ability to react to changes that might affect their decisions and considerations during use like additional people joining or a change in spatial relations like moving to a specific area or keeping certain distance thresholds.

For issues arising with data processing, Ubi-Interact is trying to move towards minimalistic interfaces between separate systems that define and demand only what is essentially necessary for interactivity instead of granting generalized access. Individual instances of open-source and inspectable processing modules should also give confidence in data ownership being guaranteed to the degree that is achievable while the range of interactions with remote systems is not being substantially limited.

The web server itself should make sure no process/execution triggered by clients can gain elevated access rights to e.g. filesystem that go beyond the client's authority and could potentially compromise integrity.

N-Dimensional Content Reasoning

From the various application scenarios and the consideration about context (Section 4.7), it is clear that for Ubiquitous Mixed Reality to work beyond a limited experimental scope it is essential to be able to take in various data along multiple dimensions to be viable. Traditionally, spatial reasoning has taken a center place and is still the main consideration in most Mixed Reality applications, but semantic ([129], [161], [130], [132]) and temporal ([131]) reasoning about events is finding more and more influence.

The design goal for Ubi-Interact is to give each element involved in the system the possibility to identify and access the relevant data for these reasoning capabilities in a convenient fashion. Security & privacy concerns need to be weighed against these options, i.e. they become part of the reasoning mechanism itself.

Interoperability, Extensibility

A major revision update in software can trigger a cascade of issues in a larger

scale system when APIs become incompatible or packages and dependencies are not (yet) available for the newer version. If a project wants to move to a new version or parts of the project require an update, it should not have the consequence of paralyzing other parts or having to restructure them. A clear separation of concerns and self-contained runtime environments for individual parts of a project can help alleviate these issues.

Conversely, in a distributed system or heterogeneous accumulation of systems interacting with each other, one should assume that individual parts will be operating on different standards to some extent. Some elements may become old and outdated or rely on older software versions. They can lag behind with updates or simply not receive any in the future. They may be replaced in hardware and software with newer, better, more efficient solutions.

Ubi-Interact is tasked with keeping inter-dependencies of involved systems low while maintaining high flexibility within a single runtime. It should pay attention to keep transitions of and extensions to existing functionality smooth and choose an architecture that will not force major updates on existing integrations. It should also help with designing software in a way that incentivizes interoperability and extendable.

Convenience

Of course, any framework trying to attract users should aim for convenience. This means opening up complex settings only if desired by the developer and otherwise hiding or properly auto-configuring them.
Integration of the framework into existing software should not pose any major hurdles. Keeping a setup running and up-to-date should also be quite convenient and not demand tedious amounts of extra work. Documentation of implementation steps and updates should be plentiful and kept accurate.

Quality Assurance

The need for quality assurance is true for the developed framework itself as much as for the framework to provide tools for developers that want to test their implementations.
The focus for Ubi-Interact is to a) supply debugging tools like inspectors and monitors for state and events of the system, b) continuously profile the system with performance and integration tests that are accessible and reproducible by users of the framework themselves and c) make modules and parts shareable between users so others may test them, adopt them, improve them and generally benefit from them instead of having to redo the work.

Swan[162] commented on the importance for the scientific process to replicate findings and affirm their validity. Testing findings and solutions under slightly differing circumstances and permutations can also help solidify results or extract and condense key aspects.

Ubi-Interact focuses on three things:
1) To be easily integrated into the native runtime of existing solutions and - vice versa - being able to quickly adopt third-party findings with their native runtime into one's own system.
2) Providing convenient wrapping and interfacing functionality exposing existing solutions and modules to a wider system.
3) Being able to easily swap similar solutions against each other and make them comparable while the rest of the system can stay the same.

In terms of system design and use, applying the principles of Poka-Yoke[163] for quality assurance can certainly help build interfaces that are not ambiguous and easily misinterpreted and therefore incidentally and unintentionally broken.

Integrability

Based on the assumptions about future Ubiquitous Mixed Reality from Section 2.2 any system operating in such an environment is only one part of a bigger puzzle. A system then faces the complementary questions of "How well can it integrate foreign systems into its own behaviour?" as well as "How well can it be integrated by foreign systems?".

Ubi-Interact takes both questions as fundamental design decisions. It must be able to provide an umbrella for isolated systems and processes but it must also be able to allow influences and a certain degree of control about its inner workings from outside and provide clear paths on how information can be exchanged and interaction can happen.

5.2 General Design Decisions

The very first intention for Ubi-Interact was to build highly individual setups incorporating multiple personal devices to form a "digital skin" or "digital suit" which would flexibly embody users in the digital world depending on the capabilities of the devices utilized and the tasks and intentions for the current digital environment - with the assumption that these setups would have to work within the context of a larger and a-priori unfamiliar and dynamically changing environment.
It would have to offer easy and efficient ways of integrating with other systems that follow their own architecture, standards and/or purposes. It would therefore also have to offer (ideally reusable and shareable) ways of identifying surrounding systems, reaching out to them and establishing communication. These would probably take the form of modules acting as mediators, translators and/or control instances for the external system.

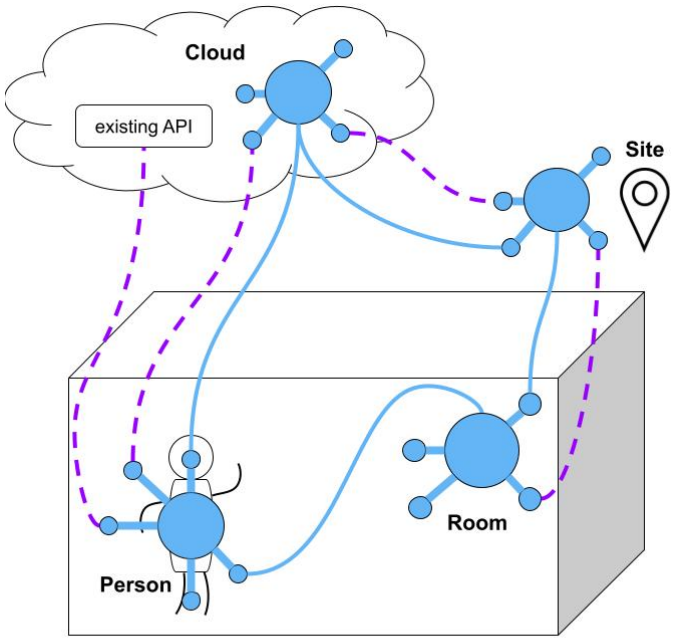


Figure 12: Ubi-Interact working on different scales, integrating with other systems. Big master nodes with smaller peripheral client nodes. Solid blue connections indicate protocols native to Ubi-Interact , dashed purple lines are foreign protocols.

5.2.1 System of systems

If such a system was effective in extending a hand and "play ball" with the rest of the world, it would be even simpler to build networks consisting of instances of the same system. These networks could be organized hierarchically/vertically, but they could also form decentralized, lateral, peer-to-peer relations. Figure 12 shows a hierarchical scenario with Ubi-Interact instances acting over several scales from a personal level up to cloud-hosted services. Ubi-Interact takes a "divide-and-conquer" approach with multiple layers of operation rather than a highly scalable single-instance approach to cover aspects of scalability and avoid massively centralized data and processing accumulation out of privacy considerations. Any of the instances should of course be replaceable with a "foreign" system as indicated by the API box inside the cloud. Solid blue connections indicate communication channels following Ubi-Interact's protocols while dashed purple connections stand for arbitrary other protocols, possibly peer-to-peer.

This general conception brings with it the implication that Ubi-Interact is trying to stay protocol agnostic to some extent. Blanco-Novoa et al.[159] also comment on the difficulties trying to rely on a single protocol when different fields of research with solutions geared towards their needs are converging.
While protocols and standards are obviously necessary to establish any communication channel in the first place, Ubi-Interact is trying to treat them as equally viable and use them in parallel as much as possible - not only towards outside communication but also with respect to their internal integration in the Ubi-Interact system itself. In consequence, this means that one instance should offer multiple parallel ways of connecting - with additional options integrated as necessary. The internal standards should not be too intertwined with the functionality apart from naming and conceptual conventions and thus leave the possibility to be swapped/alternated.

5.2.2 Centralized local units

We can see that each Ubi-Interact configuration is centered around a bigger master node with smaller orbital client nodes around it.
The master node acts as the logistics and management center, handling central services essential to all Ubi-Interact clients as well as passing messages between client nodes.
Client nodes act as the communication interface for processes and systems with the rest of their local Ubi-Interact neighborhood, transferring e.g. device data or triggering reactions to events. If multiple entities live and work in the same system process, they can rely on a shared client node.

While decentralized peer-to-peer architectures certainly have advantages in some regards (see Section 4.2.2 and [106]), a centralized architecture was chosen instead of peer-to-peer connections as the goal was to treat all possible client environments equally, especially those that are not capable or allowed to act as servers themselves and thus can not form any host endpoints of peer-to-peer connections.
Examples of this are headless IoT devices (see Section 4.2.4) or web applications running in the browser. These environments rely on centralized architectures or otherwise have to rely on special connection bridges and proxies.

In Figure 12, both the dashed purple lines and the solid blue lines connecting a client node to a remote client or master node indicate more specialized connections that are capable of building such bridges outside the conventional ways of communication.
In the case of solid blue connections, a client node establishes a special routine opening another client connection to a foreign master node and relaying any relevant information between both setups.
In the case of dashed purple connections, it is two client nodes or respectively a client node and foreign API endpoint communication over a different proto-

col. Pub/sub protocols are typically geared towards efficiently processing many small data packages and are therefore not very well suited for large amounts of data. A good example for such a connection would be a peer-to-peer audio or video stream protocol, which is highly specialized to its purpose and therefore vastly more efficient in its implementation than any of the "native" Ubi-Interact protocols. Setups may also require a bridge to another pub/sub broker, for example MQTT.

5.2.3 Architecture Goals

All this should illustrate that while a singular Ubi-Interact setup is centralized in its architecture for reasons of simplicity and equal treatment of client nodes, it is in no way ruling out specialized peer-to-peer or other communication between its client nodes and other systems.
The purpose of Ubi-Interact is not to cover every use-case, it is to bring heterogeneous distributed systems together and integrate with them. It is, however, a goal for Ubi-Interact to enable users/developers to share their implemented nodes and processes dealing with such specialized use-cases - as is very common with tools like ROS (Section 4.3.1) and Node-RED (Section 4.3.2) - so that solutions can be built, adapted and maintained more time-efficiently and with better quality.

Another question is how to connect to legacy systems that might not bring their own communication/interaction API. In that case it is necessary to establish a Ubi-Interact client within their process, taking over the job of outside communication. It also follows that client nodes must be built for their respective environments so they can natively fit into the rest of a program. As an open-source project, once a client node has been established for a certain programming language or operating environment, it can be adopted and maintained by all active users.

Two goals can be derived from this architectural decision:
1) In terms of intra-dependencies inside a node, the aim is for maximum freedom in the choice of environment (programming language, runtime, etc.) a client node is running under as well as additional software, libraries, drivers and so forth. Only that way can it be guaranteed to accommodate special cases where certain performance, hardware or software is required. The framework itself should not by default limit the developers in their choice of additional software they want to make use of.
2) In terms of inter-dependencies between nodes, the aim is for minimal dependencies between them. Client nodes should be able to connect to the master node via multiple protocols, ideally covering all commonly used ones. As the common language, i.e. the message exchange format, it must be easy to include and use by all major platforms and environments.

With nodes being implemented for their respective programming environments and being used to connect additional software processes with libraries etc. to the larger Ubi-Interact network, they easily lend themselves to also supply data processing solutions to other nodes when those solutions are not native or harder to do in their respective codebase - if the network latency and overhead is an acceptable downside.
And while the master node is capable of integrating additional processing tasks just like any client node and indeed is a natural accumulation point to access all available data, running these tasks in the same process as the master node has to be considered with special care as to not interfere with message delivery performance. If additional performance is available on the master node hardware, it is usually best to establish another client node in the same machine and rely on inter-process communication (IPC).

On a less technical but personalized level, Ubi-Interact is intended as a framework to have a flexible combination of extensions working in tandem with each other and the environment. This includes humans as users of technology as well as simpler digital systems.

5.2.4 Message schemas and internal API

Many of the conceptual objects like client nodes, the entities related to them, etc. are implemented in every node codebase but naturally have to be communicated between nodes too, e.g. a client node describing itself and its entities to the master node, thus registering the information to be queried by other nodes.

This naturally leads to many of the node internal classes and objects being reflected one-to-one by an associated schema and message object. The common message formats between nodes represent most interfaces and API of Ubi-Interact as a whole.
Since an extendable message format definition was chosen for Ubi-Interact, a desirable property for the system was for native implementation of objects to be able to naturally grow with their message formats.

It is ultimately up to each node how to handle this, but with current node implementations it was an effort to align the internal objects as much as possible with their message representation, either by saving the respective message as a property to be safely updated/modified as needed or the object itself directly reflecting the message properties.
It should be taken care that this is implemented in a way that keeps direct code dependencies to the implementation of the message formats minimal, in case it is ever deemed necessary to replace the message definition system in the future.

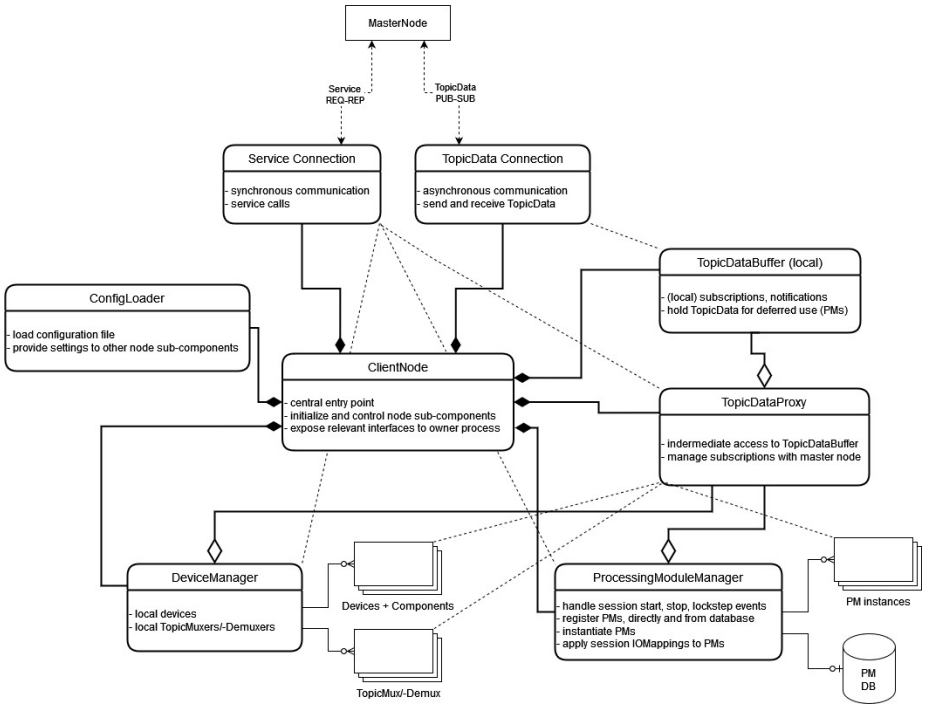


Figure 13: Overview of different elements for a client node implementation. Dashed lines indicate indirect relationships instead of direct dependencies.

5.3 Nodes

This section presents the technical buildup of nodes composing one centralized Ubi-Interact unit.

5.3.1 Client Node

For any client node implementation, there are some mandatory components and some optional ones that can be incrementally added as features.

In its minimal implementation, the client needs a service as well as event data (*TopicData*) connection to the master node. After exposing interfaces for service calls and publishing/subscribing to events using the compiled Protocol Buffers message objects, a minimal viable client node is established.

For efficiency reasons, it should also offer local subscription management as to allow multiple parties from the client node environment to announce interest in the same topic, topic regular expression or *Component* (see 5.4) without

announcing redundant subscriptions at the master node. Figure 13 locates this feature in the *TopicDataProxy*.

Depending on the node's implementation language, a *ConfigLoader* is also helpful with setup configuration like communication endpoints, encryption and so forth. Another convenient addition is a *DeviceManager* offering a simplified interface for (de-)registering *Devices* with the master node and keeping track of local *Devices* / *Components*.

All additional elements are mainly a consequence of providing a node with the ability to run *Processing Modules* (see 5.5).
Some processing modes require asynchronous access to event data, it is therefore efficient to keep these subscribed topics and their latest data readily stored in a local buffer (*TopicDataBuffer*). Once a local buffer is implemented, it is again convenient to hide it behind a proxy (*TopicDataProxy*) so other node elements relying on event data may use it as if it was the global topic data buffer on the master node. This simplifies subscriptions for the node user by removing manual service calls and callback handling for received events. Instead all this functionality is managed by the *TopicDataProxy*.

5.3.2 Master Node

Compared to a client node, the master node takes care of some global responsibilities. The major additions are thus a *ServiceManager*, a *ClientManager* as well as a *SessionManager*. It also features a *TopicDataBuffer* that is equivalent in its functionality to a client node but holds all event messages of all nodes involved.

The *ServiceManager* takes care of registering all available services.
For each incoming service request on the master node, it forwards the request to the responsible service based on the topic the service registered. Naturally, there can only ever be one service for a single topic. The service also defines what request formats it expects and which response formats can be expected in turn. Services are also responsible for formulating their response themselves and return it to the *ServiceManager*.
As such, services themselves will have dependencies on various other master node elements to accomplish their purpose - they are omitted in Figure 14 for clarity.
A master node exposes the *ServiceManager* so developers can add services beyond the standard set as required.

The *ClientManager*'s responsibilities consist of keeping track and verifying connected *Clients* as well as their registered *Devices* and *Components*. It also has an indirect dependency on the pub/sub communication as it keeps a sign-of-life status updated for each client and sends out heartbeat messages over this

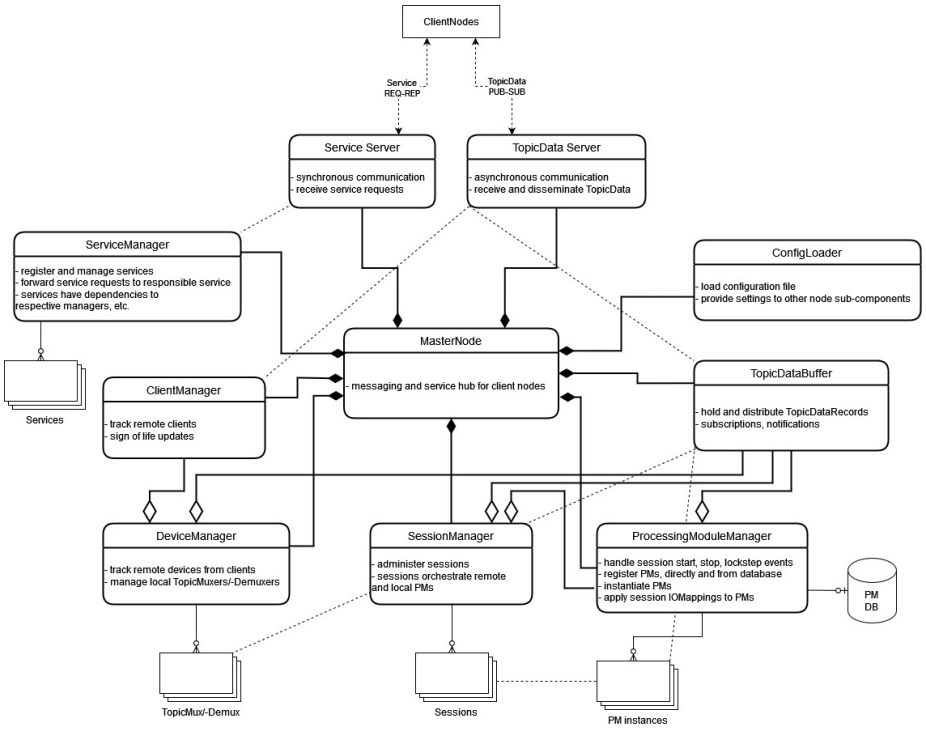


Figure 14: Overview of different elements for a master node implementation. Dashed lines indicate indirect relationships instead of direct dependencies.

channel.
Anytime the client publishes a new message, it will be registered as the client still being active. After a certain timespan of inactivity by a client, the *ClientManager* will send a "PING" message to the client, expecting a "PONG" message being sent back. If no such response is sent, the *ClientManager* will consider said client as inactive and after a second longer period elapses the client will be considered inactive or unresponsive and most of it's information will be excluded from the general communication.

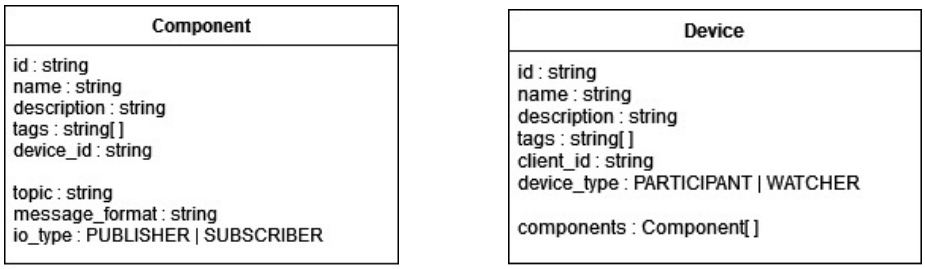
The *SessionManager* is the central control instance for all *Processing Modules*, whether they run on client nodes or the master node itself, and the *Sessions* that contain them.
It reacts to service calls for establishing new *Session* configurations, starting, stopping and pausing them. Communication with all nodes and their *Processing Modules* involved in a *Session* happens through preset topics.

5.4 Devices & Components

Devices and *Components* are Ubi-Interact's way of giving "form" to entities and specifically their I/O profile - physical and virtual - that can be understood and connected with by other parts of the system.
The mechanism of describing these profiles should 1) fit the purpose of Ubi-Interact without overextending and 2) instead of pre-defining categories allow the minimum base building blocks necessary to be extended and naturally grouped. Colloquially speaking, this is a "walks like a duck, quacks like a duck" approach to defining entities and their communication behaviour.

5.4.1 Components

Components are treated as the building blocks that either produce or consume data in some form. As such they are associated with an event data format, an event key used to identify the associated communication channel/topic and additional properties that help identify where data comes from, how it is produced or what it is being used for. In terms of abstract entities (2.2.4), a *Component* is what signals some of their properties (publisher) or expresses the ability to receive and react to some event (subscriber).
For any component, Ubi-Interact is not concerned with what happens behind it, i.e. how the data is produced or consumed. These things might inform metadata of the component but are otherwise up to the processes around nodes registering said component. It should also be mentioned that *Components* are not required in order to use pub-sub messaging, any code part can use topic-based communication directly via the node's methods.



(a) Component properties (b) Device properties

Figure 15: Component and Device properties.

5.4.2 Devices

Devices group *Components* into more complex structures. Apart from the list of *Components* belonging to them, they also contain additional meta information. Their purpose is essentially to describe this list of individual affordances to the rest of the world so others may find a tool, a target, similar individual entities and the like and thus make sense of their (virtual) surroundings.

A comment on the naming of *Devices*: There was a consideration whether devices should rather be named entities to better reflect the abstract nature (physical and/or virtual). Yet entities may carry more than just I/O data, for example intent (4.5), which may be desirable to reflect in Ubi-Interact in the future. The name *Device* was thus kept as it is also more relatable to the familiar UI abstractions of a traditional input/output device.

5.4.3 Usage

After registering a *Device* at the respective master node service, other nodes can get a list of them using a second service. If the second service is called without any additional parameters, it will give back the full list of all registered *Devices*. If however the request is sent containing a *Device* specification, the service will filter the results by matching against this profile.
The same is true for services related to *Components* and other conceptually defined elements of Ubi-Interact.
With that, any node can define the profiles they offer and also use them as searching/filtering criteria. A few examples should illustrate the concept.

In case where a device is needed that acts like a traditional mouse - maybe there even is a regular mouse, only it is plugged into another system - it could be expressed as a *Device* having at least one *Component* firing button events (possibly two, tagged with left and right button) as well as a *Component* pub-

lishing 2D float vectors (Ubi-Interact message format *Vector2*). Whether the 2D vector is used as a relative change or as an absolute position (normalized in ([0;1][0;1]) indicating mouse pointer movement should be further classified by the *Component*, for example by carrying an agreed tag like "relative" or "absolute".
There may be mice-like *Devices* with additional *Components* (e.g. a *Component* for the mouse wheel with its respective events) but if remote systems are looking for a base mouse input device they can rely on this profile to search for matches.

A virtual car might be a *Device* too, with an entire physics simulation behind it that Ubi-Interact remains oblivious to. It could feature *Components* for input like left-right steering commands and speed throttle. For output, it may be reporting it's current speed to the rest of Ubi-Interact.

A human user may rely on a system tracking their entire body as well as their eye gaze direction or even emotional face expression recognized through some image processing library and wear a bluetooth wristband that is capable of giving back vibration signals. They could register this as a *Device* representing them in different aspects to some capacity.
If technological extensions are added or changed, the *Device* profile is extended or adapted. Depending on the implementation, the node can do this dynamically by aggregating only the active *Components* currently in use.

After a request, if the service response contains only one device, we can immediately go through its *Components* and subscribe to their topics to connect whatever function callbacks we need.
If the service responds with a list of *Devices*, either the *Device* needs to be qualified further or additional criteria can be added. The *Device* profile could be extended by another *Component* exposing its location in the area so that the nearest one can be found (in consequence requiring that viable *Devices* do implement such a *Component*. Since *Components* can have *NotifyConditions* (see 5.7.2) attached, we can rely on them to specify requirements on dynamic data and describe conditions like *Devices* withing a certain range of the request sender's location.

The ID of *Devices*, *Components* or any element of Ubi-Interact is naturally an exact matching criteria but has to be known in advance when requesting it. Any other property like name, message format or tags can be used as potentially ambiguous search criteria and only their combination and aggregation makes a target specific.
This also means that any element should be described well in its profile. If the profile for a certain concept like *Components* does not offer enough flexibility for an application case, the Protocol Buffers schema can be extended with additional fields.

It was mentioned that the functional workings behind a component producing or consuming data are of no concern to Ubi-Interact as a system itself.
This functionality usually involves a lot more (pre-existing) APIs, drivers, SDKs, system calls, etc. and the description in the form of a Ubi-Interact *Component* is a simple addendum. It can still be beneficial to have common *Components* established as their own class object with their functionality as well as Ubi-Interact profile. This eases sharing and maintenance and speeds up development by providing ready-to-use elements.

One such *Component* established quite early in the development of Ubi-Interact is a smart device touch component for web apps. Given a touch area HTML element and the Ubi-Interact node, it registers the necessary HTML event listeners, normalizes touch coordinates to ([0;1],[0;1]) and publishes all touch events to the wider Ubi-Interact system. It found multiple applications and the desired interactivity between e.g. a handheld smartphone touch display and a remote system like a model viewer, VR application or video game could repeatedly be achieved with very little time effort in about an afternoon. Any such *Components* are then of course available to be combined into *Devices* as necessary.

5.5 Processing Modules & Sessions

As mentioned in Section 5.2 and 4.3, in distributed systems involving mobile and hardware-limited devices it is often convenient or even necessary to outsource computation to more appropriate edge or cloud resources.
It also helps with modularizing solutions and offering them as a black-box service to a wider number of clients. Sometimes these modules need additional setup and orchestration with the rest of the distributed system in order to function in time.

Processing Modules are the interface to describe these black-boxes to the Ubi-Interact network together with configuration on how to operate them so others can find, instantiate and connect with them as required.
The idea is not to have a static number of instances of a module but to allow instantiation as required. Depending on computing resources and application case, a single *Processing Module* may handle all entities and data - especially when they need to be processed in relation to each other - while in other cases individual entities may rely on separate instances of the same module for their subjective needs and states.

Sessions are a way of giving the *Processing Modules* a runtime configuration without the modules having to know anything about specifics like other *Clients* and their *Components*.
Sessions arrange which data channels are actually connected to the *Processing Module*'s inputs and outputs. They provide the wiring of *Components* to *Pro-*

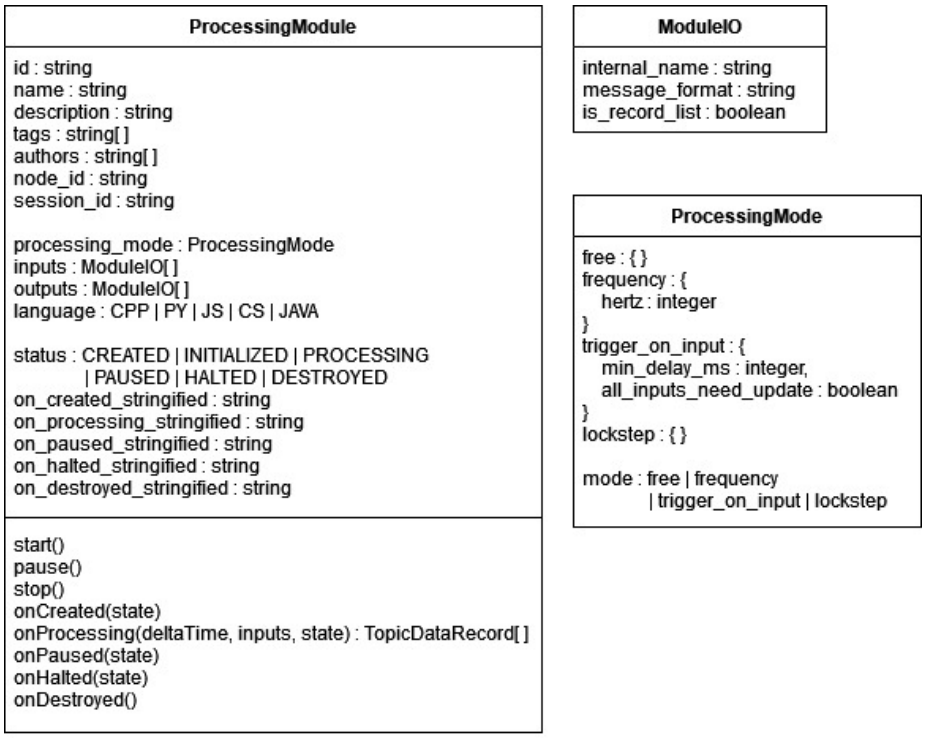


Figure 16: Properties of ProcessingModules and related objects

cessing Modules so either can stay truly modular and be implemented without prior knowledge about the rest of the overall setup. A *Session* is also designed to give an enclosed space for *Processing Modules* to work in, possibly including and excluding other modules and clients and providing synchronization points specific to this *Session*.

5.5.1 Processing Modules

Processing Modules[46] are designed with continuous event processing in mind and less about request-reply-like communication. Using the *trigger_on_input* mode described later and a dedicated setup of responsible client *Components*, one can turn them into a module with remote procedure call (RPC) characteristics. Still, a more dedicated processing mode together with the master node acting as a service request and reply proxy for the *Processing Module* would be better suited for these use-cases and may be implemented in the future as necessary.

[46] `https://github.com/SandroWeber/ubii-msg-formats/blob/develop/src/proto/processing/processingModule.proto` (last accessed 08-06-2022)

The base structure of a *Processing Module* consists of 1) some general descriptive data, 2) the I/O and processing profile with inputs, outputs, processing mode, implementation language and 3) status and lifecycle with callback functions (see Figure 16).
Apart from the general information like *id*, *name*, etc. and what node and *Session* it is running under, the core continuous functionality provided by a *Processing Module* is implemented inside its *onProcessing()* function which is further described in its parameters by the *inputs* and *outputs*.

inputs & outputs
A module's *inputs* & *outputs* are specified in the form of a list of *ModuleIO* objects. Their *internal_name* is the variable name used by the module's author(s) to address its I/O within the lifecycle callbacks - this ensures a module can be implemented without coupling it to external parameters. The *message_format* describes the type of events that are expected and *is_record_list* describes whether the input/output actually refers to a single event record of type *message_format* or a list of records all of that same type.

processing_mode
The choice of a *processing_mode* determines when and how the central lifecycle callback of *onProcessing()* is triggered. As a central piece of control over how the *Processing Module* is working together with the rest of the system they deserve a detailed explanation:

- free:
  The *free* mode allows the module itself (respectively the process it is running under) full control over when to call its *onProcessing()* function. As such it is the most asynchronous mode of execution, there are no outside checks, guarantees or control on timing. It is especially suited for cases where Ubi-Interact can or must not have any say in the execution cycle. An example for this is a module where functionality is entangled with a physics engine which has its own strict update cycle that needs to be adhered to.

- frequency:
  With the *frequency* mode starting the *Processing Module* will establish a task calling *onProcessing()* in a set time interval (with best effort). This mode still operates quite asynchronously as it does not react to input/output events and will take the latest data present at the node's buffer whenever *onProcessing()* is called. It is best suited for cases where continuous and frequent updates are required but the synchronization with inputs and outputs is of no concern and best effort on the latest available data is enough.

- trigger_on_input:
  With *trigger_on_input* mode the module can react to input events and

only process if necessary. For more fine-grained steering of execution, the *min_delay_ms* can set a minimal delay in milliseconds between execution - this can e.g. prevent a flood of execution triggers with inputs that occur in rapid succession or adjust execution to available computational power. The *all_inputs_need_update* setting requires that all input channels of the module be updated since the last iteration before triggering another one. Future extensions to this mode could include a selected list of inputs (referring to the *internal_name*) that act as execution triggers.

- lockstep:
  Lockstep is a synchronized mode where the overarching *Session* controls exactly when each processing iteration happens. Inside one *Session*, all *Processing Modules* following this mode will execute in lockstep and the next iteration will only happen after all modules have finished.
  The synchronization of input and output happens by the *Session* on the master node. Before each processing iteration, it will gather all relevant inputs for all modules involved. This of course means that inputs are in no way hardware synchronized between nodes producing the input data and the node executing the *Processing Module* - it is only guaranteeing that all modules will process on the same data snapshot in time taken by the master node. The *Session* will then send processing commands to the respective nodes of each module with that command message containing the needed set of inputs from the snapshot. It will then wait for each processing node to report back with an appropriate response containing their respective outputs. After all *Processing Modules* have finished and their outputs have been received, the outputs are written back to the global event data buffer by the *Session*. At this point, the next cycle can be started.
  To reiterate, this does not involve any event timing or hardware synchronization between producing input data and processing it. It only guarantees that two *Processing Modules* running under *lockstep* in the same *Session* will see and work on the same timeset of inputs and their outputs will be written back simultaneously.

language

Language simply indicates which language the module was implemented in. This is useful in cases where two versions of a module with the same purpose are to be implemented in different languages and compared against each other to evaluate e.g. performance.

status

A module's *status* gives information about its current state (see Figure 17).

lifecycle callbacks

Figure 17 shows an overview over a module's lifecycle and associated functions. The functions of *start()*, *pause()* and *stop()* are accessible from outside and

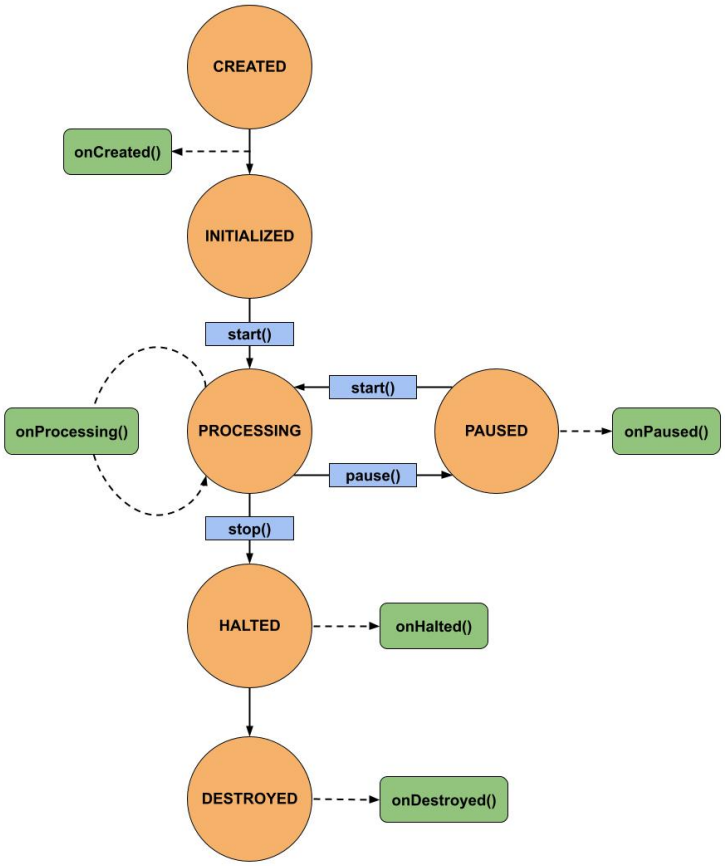


Figure 17: State and lifecycle of *Processing Modules*. Orange circles are states, blue squares are execution flow commands accessible from outside and green rounded squares are automatically triggered event callback functions.

can be used to steer execution. The callbacks of *onCreated()*, *onProcessing()*, *onPaused()*, *onHalted()* and *onDestroyed()* are automatically triggered through their respective events in the lifecycle of a the module. A *Processing Module* will only reach its INITIALIZED status after *onCreated()* was executed successfully.

Regarding the signature of lifecycle callbacks, all of them - with the exception of *onDestroyed()* when the module object itself already stopped existing - take the module's internal state as parameters. This is intended to access the internal state for setup and teardown operations.
For *onProcessing()* specifically it also takes the time passed since the last call and its required list of *TopicDataRecords* as inputs. It then returns its outputs, again a list of *TopicDataRecords*. With the execption of the *free* processing mode, this means before each run of *onProcessing()* the task controlling execution will gather all inputs from the local *TopicDataBuffer*, map the *TopicDataRecords* to their internal names for easy access and provide them as the "inputs" parameter. Conversely, after *onProcessing()* is done, the set of returned *TopicDataRecords* mapped to their internal output names is accepted and published to their respective topics.
While any of these parameters could theoretically also be accessed through means internal to a class object like a private getter/setter, it makes the working structure of a *Processing Module* more explicit and - depending on the node's implementation language features - offers the possibility to execute calls on a thread/process worker pool where input/output parameters of functions may be required to be serialized through message marshalling.
Given this reasoning for an explicit functional signature, it is not forbidden for node implementations to offer getter- and setter-like structures allowing this internal retrieval and writing of values through their internal names as variables, even while *onProcessing()* has not finished. This more dynamic access means that within the *Processing Module*, a getter under the name of its inputs' internal names would pull the data from the node's local *TopicDataBuffer* only when actually needed during processing. It also allows for outputs to be published during execution as an intermediate update or simply a faster result whenever an output variable is written via its setter, indirectly publishing the provided *TopicDataRecord* under the mapped topic.

The properties ending in *..._stringified* are an option for clients to transfer *Processing Modules* in their entire functionality to the master node and have them run there. This is only possible for stringified Javascript functions. It can be used for very simple self-contained functionality with only base dependencies. It is otherwise not recommended and can pose security risks, so an option in the master node configuration can prevent this type of specification from being used.

As mentioned above, *Processing Modules* could be specified and transmitted entirely in Javscript via the respective (message) properties. The more flexible and powerful way of implementing modules however is to write them as their own

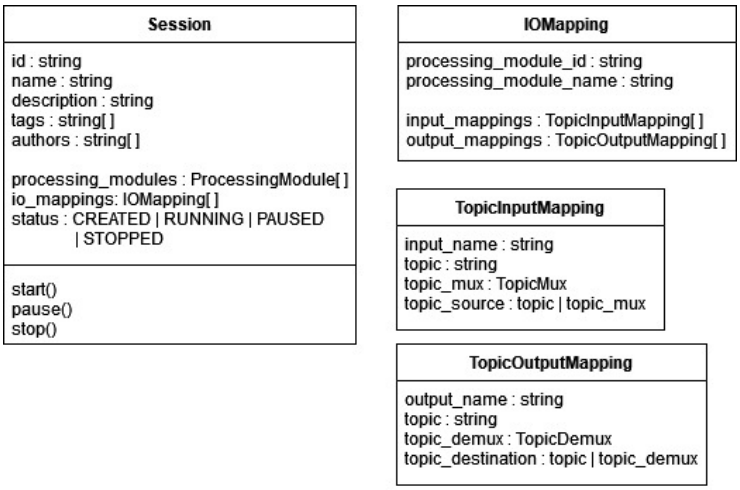


Figure 18: Properties of *Sessions* and *IOMappings*.

class that inherits from a base *Processing Module* class, implements an according interface or simply allows to pass a corresponding message specification to its constructor as well as generate a message object from an existing instance. That way an author of a module is completely free to choose any additional dependencies that the language or system has to offer.

5.5.2 Sessions

As seen in Figure 18, a *Session* contains a list of *processing_modules* it manages. When it is started/paused/stopped, it will request all *Processing Modules* to do so accordingly.
The most essential part of a *Session* is the *io_mappings* which determine how a *Processing Module* - identified via its ID or name - has its inputs and outputs connected to topic sources and destinations. In their current state, this can be either a discrete topic or a TopicMultiplexer/-Demultiplexer which will be discussed in their functionality later. If the *Processing Module* can not be uniquely identified via its name within its *Session* then the *Session* will throw an error message at the master node during initialization. The master node will not be able to create a *Session* with configuration ambiguity or errors and in case a client node is trying to set up such a *Session* it will be notified of the error.

Currently, *Sessions* are only about organizing *Processing Modules*. In the future these will be expanded to include access management layers for topics, probably involving a list of participating clients, devices, and so on.

5.5.3 Comparison with other data processing architectures

For users that are familiar with other prominent systems, a few comparisons to equivalent approaches or respectively ways of implementing certain patterns with Ubi-Interact should be helpful.

ROS

In Section 4.3.1 we've seen that ROS offers different ways of triggering execution on nodes within the network.
ROS's asynchronous mode would be equivalent to Ubi-Interact's *free* or *frequency* mode, depending on what is more convenient for the node's implementation system to control the execution.
The mode of operation for ROS *Actions* would be best matched using the *trigger on input* mode, with intermediate status updates reporting before the *Action* is performed completely requiring that the node and *Processing Module* implementation allow direct access to output variables during *onProcessing()* (as described in Section 5.5.1, lifecycle callbacks).
Client nodes offering synchronized *Service* structures is not possible in Ubi-Interact yet. However, it is possible to add extra services inside the structures of the master node.

Node-RED

When compared to Node-RED, a Ubi-Interact *Processing Module* can most closely be compared to functional nodes executing arbitrary Javascript code in their onMessage() configuration. A *Session* then resembles the flow containing a configuration of nodes that can be started and stopped as a whole.
Node-RED nodes configured to have inputs are triggered whenever a message arrives. This is how Ubi-Interact's *trigger on input* mode operates. However, where Ubi-Interact defines separate modes of execution, Node-RED simply provides additional nodes like "trigger" that can fire in set intervals and whose output messages can then serve as input triggers to the next node - this achieves the same result as a *frequency* processing mode. The execution triggering event then represents a message within the system itself, something that has not been requested for Ubi-Interact yet but might be useful to know.
In order to save states between execution, nodes in Node-RED can also rely on a context objects covering different scopes of the system like node, flow and global. In Ubi-Interact, any state beyond the scope of the *Processing Module* itself needs to be shared via topic communication or other outside means.

In Node-RED any functionality working on message events is encapsuled in a node. Especially small-scale operations that could be seen as pre- or post-processing steps for major functionality within the pipeline like aggregating, filtering, rerouting, multiplexing, etc. lend themselves to be templated and reused often.

Ubi-Interact here faces the additional complication that steps of the pipeline may be distributed over the network. It would be inefficient to establish pre-/post-steps on another client node than the main functionality itself as it increases bandwidth and latency. One potentially beneficial place to put such functionality would be on the master node itself. Right now, Ubi-Interact takes the approach of meta-devices created on the same client node as the *Processing Module* that do the preparatory or follow-up work up- and downstream of the *Processing Module*'s inputs and outputs. Further investigation on this topic is postponed to future work.

5.6 Choice of Implementation Language

Any client node is obviously implemented for its respective environment.
For the master node, the main considerations are I/O and networking performance as well as enough flexibility to cover a wide range of connection protocols. Quick iterative development cycles are a big bonus in the early stages of testing and research. For the reasons given in Section 4.9.2 Node.js is a good candidate and was therefore chosen.

Depending on future requirements of scalability and performance, it may be beneficial to port the master node / core message broker functionality to another language or exinsting codebase at a later stage - for now Node.js has proven to be quite sufficient in its performance, especially considering Ubi-Interact's focus on smaller individual setups organized into layers of communication instead of a centralized global server (cluster) infrastructure.

5.7 Publish/Subscribe Broker

Section 4.2 lists a choice of well-established publish/subscribe broker implementations already available. The pub/sub way of delivering messages seems well suited for Ubiquitous Mixed Reality systems due to its capability of decoupling participants, which is a necessity for dynamic environments.

Somewhat missing in pub/sub approaches is the case-by-case decision of who is allowed to receive messages. Some of the option for targeted/exclusive communication is lost or made more difficult with pub/sub. In Ubiquitous Mixed Reality , it is not a matter of participants either communicating with each other in general or not. Instead it is more of a nuanced decision. Some data channels are meant for one recipient specifically, but that recipient is chosen in the moment and may change over time. Some messages might be relevant within a certain range or group that, again, may evolve over time. Other messages might be for system communication and services only.
One solution is to put the decision-making into the hands of the broker which

is then (and in extension the central overarching platform) responsible for the proper communication.
Ubi-Interact is meant to enable the participants themselves to make their choices.

5.7.1 Existing Solutions

The solutions presented in Section 4.2 are topic-based. In cases of general solutions like MQTT it is clear why they would stay agnostic to the content of delivered messages.
With purely topic-based brokers, one can rely on topics as strings and regular expressions on these topics to identify data channels and subscriptions. A subscriber has to know the string - or a regular expression fitting that string - in order to establish communication. Yet the topic alone as a string has obvious limitations in its descriptiveness. A character string alone is not well suited as a schema to encode a lot of additional non-sequential characteristics, metrics, categories and relations of the data being sent.
While it is an efficient and human-readable categorization of data channels, if the environment with its topics is unknown and the notification mechanism is supposed to react to changing circumstances in the environment, the system has to provide a content-based approach that allows the consideration of topic data content for subscriptions and notifications.
In a Ubiquitous Mixed Reality case, "hard-wiring" the communication against topic strings does not cover the characteristics of our environment(s).

To further illustrate, the next paragraph discusses some of the strategies and complications of regular topic-based approaches for Ubiquitous Mixed Reality, but may as well be skipped if the reader is already familiar.

With topic-based brokers, the topic string is often treated as a strategy to establish some context for the data being communicated. Hierarchical structures can be given by the URI structure of topics, e.g. a smarthome with topics like "home/living-room/ceiling-light/intensity". In that sense it is comparable to a global variable name and implies a certain naming scheme for topics that participants know, adhere to and can be fit into. This is in and of itself not a problem, but it establishes certain preconditions on the topic string that are not enforced by general-purpose solutions while potentially causing problems if not adhered to.

If there are multiple agents that behave similarly or have to be treated interchangeable on a certain level - i.e. they all provide the same profile of topics and data like a swarm of small robots in a factory hall, a crowd of users or individuals entering and leaving a place as they see fit - the usual solution is to include some form of unique ID in their topic to guarantee no conflicts between topics and a separation of concerns avoiding their data being mixed together. So each topic might be prefixed with a UUID which then inevitably

requires a subscriber to rely on regular expressions for a topic as the UUID is or should not be guessable. Depending on how many layers of abstractions are supposed to be represented by the hierarchical structure of the topic, this may result in a rather lengthy concatenation of UUIDs.

If on the other hand every agent published on a shared single topic and included their ID with the data, subscribers would loose the ability to express interest in a specific agent and instead always receive data from everybody. Here we either face a further additional complication to the topic strings or the requirement to provide content-based filters in order to provide specific connections.

How to identify the data format being communicated over a topic is another question that typically arises. The topic could for example be suffixed with ".../int", ".../boolean" or ".../image". With this a subscriber can understand how to interpret data or look for specific types of data that may be of relevance to their intentions (another addition/convention to the topic string). Another solutions is to provide a sort of wrapper format for topic data that includes information about the contained information. This however does not allow the system to identify topics based on their message format unless a subscription is established first and data is received - which is highly inconvenient to provide contextual information about the data. Once data has been published the broker knows and can supply information, but in cases where a participant is opening a subscription in advance as a data sink accepting commands for others to publish to, this approach fails because the expected message format has to be supplied to potential publishers beforehand. This meta-information on topics is then often supplied by other means.

The data format alone might not be enough to put the data into context though. A channel for ".../image" may be addressable without producing errors, but information about the origin, content and purpose of these images is still missing. The hierarchical information of the topic may give some additional hints. Additionally, other topics belonging to the same entity providing the images might be relevant for context which in turn need to provide some context or embed themselves into the same context as the image topic.

Also affecting performance, if the topic is included with the data during transmission in its original data type of string instead of being hashed to a number, the longer a topic string becomes the more bandwidth will be consumed. This is especially problematic for small but high-frequency data like positional updates where the topic itself with all the additional necessary extensions would provide the predominant part of the message size after a certain character length is reached.

As an alternative to using the topic string itself one could instead have a convention of topics communicating context for other topics, i.e. for any topic there

is a (list of) topics to the like ".../info", ".../context" etc. providing additional information. This again imposes a convention on the topic string, which would imply certain topics not being viable - a restriction that existing topic-based brokers usually do not provide for and would instead require users/developers to adhere to by themselves.

From personal experience, this use of topics is fast and easy but does not scale so well to multiple independent users, developers or live agents extending a system because it requires a lot of implicit knowledge and familiarity. It is perfectly acceptable for networked systems with limited scale and/or a closed circle of participants like a smarthome, an experimental setup, limited industrial setting or when data exchange happens primarily between a central instance and isolated participants. It is not so well suited for open, not pre-defined environments where it is mainly about communication between unrecognized participants that may not trust each other from the start, have to familiarize themselves with their environment first and/or stay oblivious about each other while still trying to act in the same environment. It is very much applicable if systems are seen as isolated islands with very specific connections to the rest of the world, but often enough leads to unintended conflicts and difficulties in configuration and adaption for open-ended environments.

5.7.2 Broker features for Ubiquitous Mixed Reality

As [142] illustrates, security and privacy concerns in Ubiquitous Mixed Reality scenarios are sometimes more dynamic than a one-time evaluation of "Is participant P authenticated and allowed to receive data on topic T?" upon subscription by P with the consequence of P receiving all updates on T in the future. As an example, participants may in general (not) be allowed to receive data about T except when P enters or exits certain areas. In general, a Ubiquitous Mixed Reality system may need to evaluate the relations between participants, topics and potential (meta-)information about them with each update on the topic.

With Ubi-Interact one of the core ideas is to support the required context-awareness and content reasoning stated in Section 5.1 on the level of identifying communication channels and message notifications themselves. The broker should be able to dynamically react to the environment for efficiency and security reasons.
With the aforementioned scenario in which the location and positional relationship between participants matters as context, this would enable the system to judge whether it makes sense or should be allowed to communicate information and updates based on whether participants are within a certain distance of each other, can visually see each other or are residing within a certain area.
The examples of [21] and [140] (also see Section 4.7) illustrate why this makes sense and can save bandwidth and overall performance in certain situations. The same mechanisms can equally be used to implement security checks - for

example data may only be shared with clients nearby. In security cases the conditions should obviously not rely on data provided by the peers themselves as it could easily be faked by the respective client in order to fool the system.
This conditional message brokering should be able to consider static characteristics of publishers and subscribers like certain I/O profiles or belonging to a certain session or user group as well as react to the content of messages themselves without negating the benefits of publish/subscribe systems, namely the decoupling of participants.

One of the goals for Ubi-Interact was therefore to build additional enabling features into the base of the message brokering. Establishing subscriptions in the first place also brings additional complications with it if one starts from a mix of environments that has to be understood and explored first in order to generate a picture of what is available and of interest instead of having implicit assumptions about who or what can be expected to be present and which parts of the distributed system need to communicate with each other. The subscription mechanism itself must be linked to available information and relations.
The context for this conditional publishing/subscribing thus includes both static information about data producers/consumers like profiles of components, devices, clients, etc. as well as dynamic topic data like positional information.

The eventual choice for Ubi-Interact was then to build its own message broker to test these experimental features instead of building on top of an existing code-base that may not work well in supporting the planned features. An effort is made in the development of Ubi-Interact to keep topic-based and content/context-based mechanisms as separate as possible as to be able to potentially more easily port finalized and tested Ubi-Interact features to established broker implementations in order to benefit from their experience, features and performance.
Ubi-Interact's broker separates the concerns of topics as a pure data channel identifier and meta-information about the data being communicated over said channel. The result is a broker that is in its fundamentals topic-based but can include additional content-based features, enriching subscription and notification logic based upon additional info from Clients/Devices/Components as well as topic content itself. The topics can then boil down to a random UUID and serve as a (short) random identifier only.

To give a little more detail on a possible scenario where location and pose data is relevant for data communication, one can envision a VR offfice/conference environment with multiple rooms and digital avatars that represent their users with high fidelity down to individual finger movements. Any user may freely publish their body pose information, but if users are not directly looking at each other or are even located in different rooms, it may not be good use of bandwidth to propagate all updates to all other users at all times.
Instead users could express interest in body pose updates with the added condition that peers would be located in the same room as them or within their

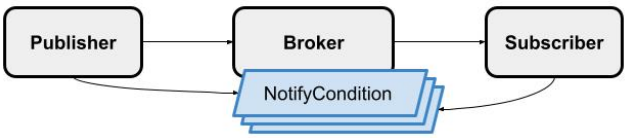


Figure 19: Both publisher and subscriber can formulate conditions without knowing specifics about each other. Any potential notification is evaluated for each publisher-subscriber pair if necessary.

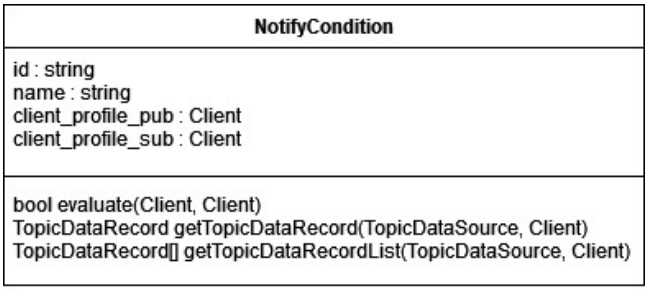


Figure 20: NotifyCondition API - general conditions on the profiles of publishers and subscribers can be set via the client_profile_pub/...sub option. During calls to evaluate() the client profiles of publisher and subscriber for which it is to be evaluated are passed as parameters and methods to retrieve topic data are available.

approximate field of view. Another condition for specific body parts like fingers could be to have a maximum distance away from them in order to receive updates because finger movements are deemed indiscernible and irrelevant beyond a certain distance. Those same conditions could also be imposed by the publisher instead of the subscriber, or both having different conditions that make sense for their purposes.

Neither publisher nor subscriber have to receive and process the opposite's position in advance, they only establish logical conditions for notifications on the master node which naturally has access to data from both sides. The decoupling of pub/sub as a central requirement is thus kept.
These conditions may of course include the requirements for other clients to provide the necessary information in the first place or have other prerequisites filtering out potential peers before a subscription can even be established.

The idea is for both publishers and subscribers to be able to formulate conditions on message dissemination/reception without having to know their peer and/or exact topics containing data that is relevant to the evaluation of the condition. The NotifyCondition API (see Figure 20) is thus passed the client profile (which includes Devices and Components) of both publisher and sub-

scriber for each notification. The API also allows to retrieve topic data based on the topic itself, a regular expression or a Component profile. While a topic is specific in and of itself, regular expressions and Components may be more ambiguous, therefore the topic data retrieval can be further qualified by a client profile to restrict which sources of topic data to consider.
If, for example, a component profile is provided that may fit for several clients but the topic data to consider should come from the subscriber specifically, the addition of the given subscriber client profile provided to the evaluation call will limit the search of components to the subscriber client only. If a single topic is expected, the use of getTopicDataRecord() will ensure an error is thrown in case multiple components fit the given profile. In the other case, getTopicDataRecordList() is expected to give a list of topic data for all fitting sources for the given regular expression or component.

The evaluation happens within a JS method expected to return a boolean, making it possible to use common math, mapping, filtering and similar functions available in JS to formulate the behaviour of the condition. ... getclients(profile, topic data)

5.7.3 Performance Measurements

The performance characteristics of the message broker are naturally relevant for a real-time system.
To get a better picture, a series of performance measurements have been conducted under different hardware configurations and protocols. The full set can be viewed in Appendix A.1.

These measurements are in no way meant to indicate a competitive performance. Rather they serve as a means to get an approximate picture on the viability of Ubi-Interact for its intended purposes. They also serve as a basis for comparison for future performance measurements. Lastly, they can give an indication on where to improve Ubi-Interact's broker performance or compare the viability/necessity to move to a different broker at a later stage.

The following hardware was used to run the tests:
A) AMD Ryzen 5 3600 6-Core Processor (3.60 GHz) CPU, 16GB RAM under Ubuntu 20.04.5 LTS
B) Raspberry Pi 4 Model B under Raspbian 11
The router in all local setups was a FritzBox 7430.

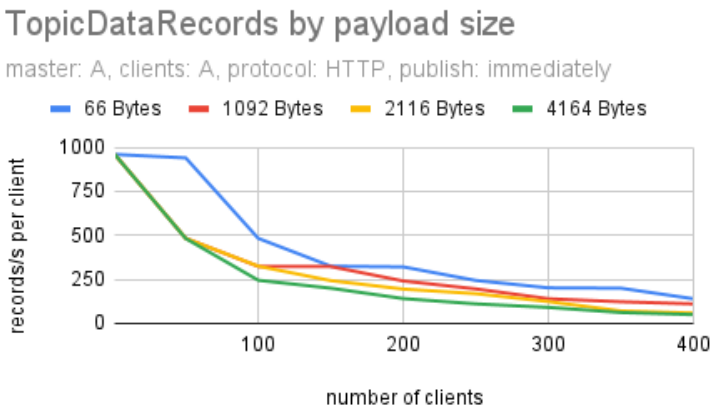


Figure 21: Broker throughput of *TopicDataRecords*. Graphs are split by record payload size. Protocol used is HTTP/WebSockets.

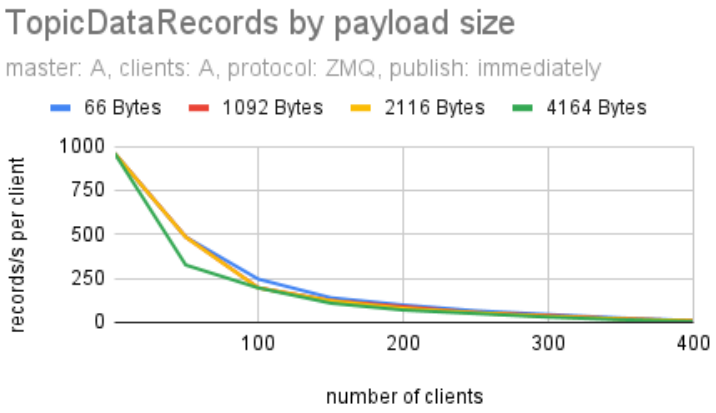


Figure 22: Broker throughput of *TopicDataRecords*. Graphs are split by record payload size. Protocol used is ZeroMQ.

Figure 21 and Figure 22 show a sequence of measurements comparing overall throughput of *TopicDataRecords* per second depending on the number of simultaneous registered client nodes and the payload size per record.
Each client node ran under a separate NodeJS process. Both ZeroMQ's TCP socket connection and HTTP/WebSocket connections were tested. All nodes, the master node running the broker as well as the client nodes, ran on the same system (A) connected via localhost.
The overall number of *TopicDataRecords* is evenly distributed over all clients.

The method of publishing was to send out each *TopicDataRecord* separately and immediately instead of bundling them in set intervals - this puts maximum load on the connections, the bundled method typically increases performance.

The client nodes were configured to send out a *TopicDataRecord* on an individual topic and subscribe to their own topic. Consequently each client would receive back each record published, allowing time delay measurements and effectively doubling the amount of messages communicated.
The records consisted of a timestamp (64 Byte) that was used to measure time between publishing and subscription callback being called. Additionally, each record carried a random string payload of variable size - 0, 1024, 2048 or 4096 Bytes. This leads to the overall record sizes of 66, 1092, 2116 and 4164 Bytes for the serialized Protocol Buffers message bytestream.

Each test was run with constant record payload and constant records per second per client. The number of clients were set to ramp up over time to the set limit. After the target number of clients was reached, the test was continuously run for at least five minutes to allow for a settling in phase.
Additionally, the master node is running a profiler. It was configured to check overall performance every five seconds and to print console warning statements in case performance was affected (elapsed time since the last gathering of statistics being at least 10% delayed, in this case more than 5.5s).
If the profiler continuously issued multiple warnings for degraded performance over one minute or the number of maximum active clients was not continuously upheld (i.e. some clients were considered inactive from time to time until the broker could catch up), this would be regarded as unstable and unacceptable performance for the test.
A test configuration that would overload the broker would typically be detected within seconds to half a minute after reaching the set targets. For these reasons, five minutes was determined a reasonable stretch of time to settle into a stable state of message handling.

The outer boundaries of the graph require some additional contextualization. Each NodeJS client node process was able to handle no more than approximately 960 *TopicDataRecords* sent and received per second (1920 total in + out). For this reason the graph is capped for one client at slightly below 1000 records per second.
As for the right boundary, 400 simultaneous NodeJS processes was the maximum the quoted hardware (A) was able to handle.

Figure 21 clearly shows a degradation of performance with increasing number of clients. Interestingly, HTTP/WebSocket shows better performance overall than ZeroMQ/TCP - this may hint at a suboptimal integration of ZeroMQ libraries under NodeJS but could be due to other factors as well and should be investigated further.
The performance down-trend is also true when broker and client nodes are run

on separate hardware, as seen in Figure 23 where the master node runs on a Raspberry Pi 4.

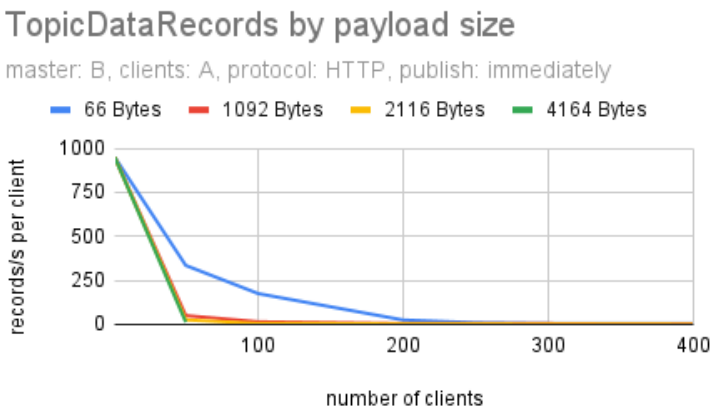


Figure 23: Broker throughput running on Raspberry Pi.

Naturally, the Raspberry Pi (hardware B) offers little performance compared to hardware A. It is nevertheless usable if the number of clients stays below 150 and payloads are small on average.

Round-Trip-Time (RTT) has also been measured. While latency measurements are of interest in general, they depend on the infrastructure and should therefore best be performed in their live network environment (more later).
On localhost, measured latency is below one millisecond. An average of 0.38ms with a minimum below 1ms and maximum of 2ms was recorded.
Over WiFi, the average lies at 6.28ms, minimum recording being 2ms and maximum 335ms. RTT was also tested for a deployment on a university virtual machine (VM).
From within the university network, RTT was again quite low: average 0.85ms with a minimum of below 1ms and maximum of 5ms.
Latencies were also measured over the internet from a site roughly 18km away. In an earlier round of tests, the average was determined to be 23.18ms (min. 20ms, max. 28ms). Interestingly, after a restructuring of the university network in June 2022 the performance dropped to an average of 34.48ms (min. 33ms, max. 37ms).
This observation reinforces the notion above that such performance tests should always be reproduced and run in their actual environment.

Appendix A.1 also includes graphs for overall bandwidth throughput. The maximum bandwidth observed was just below 280MB under hardware configuration A, HTTP/WebSocket and the bundled publishing mode.

5.8 Debugging

The requirements of convenience and quality assurance are crucial aspects for the development of distributed systems.
In practice, this requires effective ways of debugging during development as well as live usage. Ubi-Interact supports this via several mechanisms.

All communication has the option to report errors. Service responses can always include an error report related to the preceding request. In case of asynchronous occurrences of errors, any *TopicData* message may also include error data. As nodes operate in different environments, it is up to the node and its surrounding process to decide in what fashion to convey these errors (console, GUI, etc.).
Of course, the option to open dedicated topics for error reports exists as well. There are no mandatory or default topics for these cases as general report mechanisms exist in the way described above, but any setup may establish its additional error reporting mechanisms according to its needs.

Tracking performance and identifying bottlenecks and breakdowns is another important feature. *Nodes* are continuously extended to provide performance and latency measurements for their message publishing and execution of *Processing Modules* as far as they are expected to run according to a specified performance. For *TopicData* - as there are no acknowledge messages passed back from the master node - this is only applicable to the bundled publishing mode that gathers records and flushes them in set intervals.
For *Processing Modules*, all execution modes except for the *free* mode are expected to follow some form of control, be it reacting to a trigger or executing according to a fixed schedule. This again allows to measure elapsed time until execution of a processing iteration is finished.
Additionally, the master node keeps track of all client node latency through heartbeat messages. It will update their status as active, inactive or unavailable after certain delay thresholds have been exceeded.

Graphical tools for debugging are available in the web frontend and will be presented in Section 5.10.

5.9 Testing

Equally important to cover quality assurance, testing should be a constant endeavour during development.
With distributed systems, much of the functionality is reliant on communication and behaviour may change with live runtimes. Ubi-Interact thus focuses more on integration tests that can be run and reproduced by any user/developer. Base functionality of services and the message broker are unit tested, but the

performance can only really be evaluated in its actual deployment, as it heavily depends on network infrastructure and hardware.

Consequently, Ubi-Interact's performance is constantly tested against previous developments in the same environment(s) to ensure it does not needlessly or unintentionally degrade. All tests - most of them available through the web frontend (Section 5.10) - are available to be run through a GUI or a simple script with help instructions.
Deployments and test runs are also performed over longer periods of time (hours) under load to identify potential stability and performance failures like ramping memory consumption or building up message processing queues.

At the same time, these integration tests can serve as a code example on how to use certain features. Tests exist for simple publish/subscribing or the setup and use of a specific execution mode for a *Processing Module*.

5.10 Web Frontend

The web frontend developed for Ubi-Interact serves as an introspection, debugging, administration and testing UI. To this end, it offers tools, integration tests and exemplary applications that can be run from the web GUI. The following will quickly present only some of the tools deemed helpful when building distributed applications with Ubi-Interact.

From a technical perspective, it acts like any other client node - utilizing the node developed for browser environments. It is usually deployed on the same hardware as the master node but can in principle be served on any other machine too.

5.10.1 Client-Device-Component Viewer

With this tool, one can see a list of all *Clients* with their registered *Devices* and *Components* (see Figure 24).
Clients that have not responded to a heartbeat message from the master node for a while are categorized as unresponsive after a certain timeout. This may include *Clients* that have disconnected without deregistration and will be listed separately as to not clutter the active list.

5.10.2 Topic and Service Inspector

One of the tools included in the frontend is a topic data and service inspector (Figure 25).

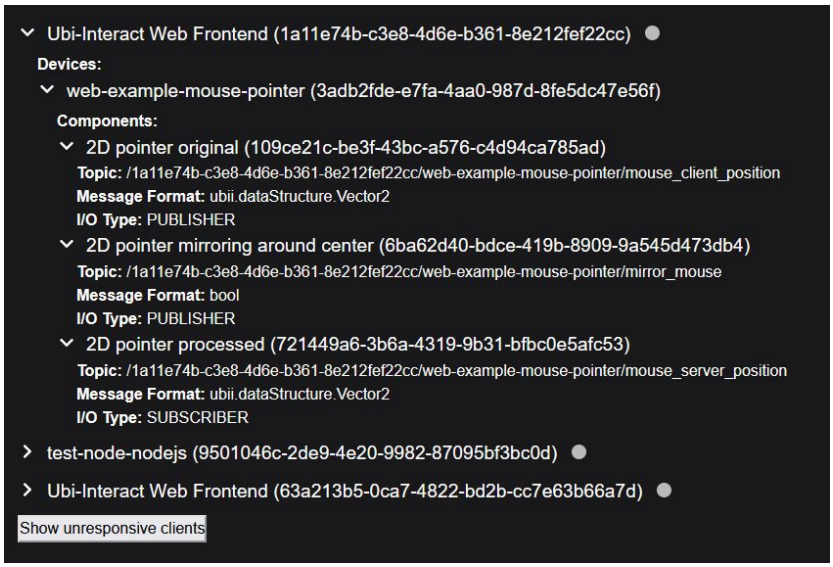


Figure 24: View *Clients* with their registered *Devices* and *Components*.

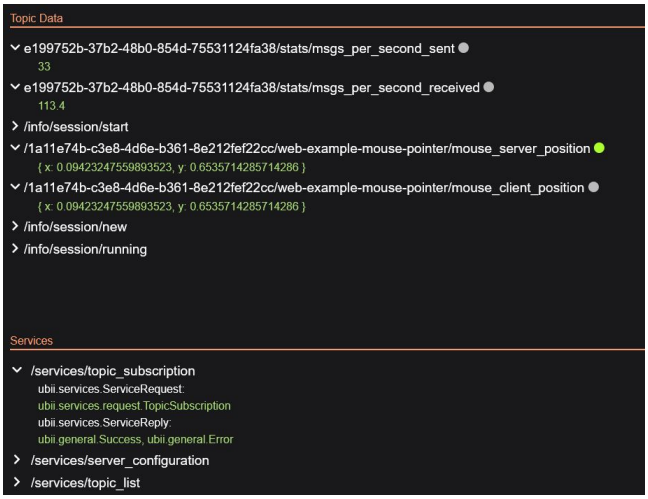


Figure 25: Tool to view topic data and available services. Topics will be receive live updates (green light next to topic name and their data can be inspected. Services include their request and reply message format(s).

Figure 26: Write *TopicDataRecords* in their JSON object notation and publish them.

In the category topic data, an updated list of all topics is available. Once a topic is expanded, the inspector will open a subscription. Consequently its data is visible and will receive live updates. Only expanded topics are subscribed, as a general subscription to all topics can quickly lead to overload depending on the amount of data communicated through the rest of the system.
This tool should still be used carefully, as many expanded topics equal a lot of subscriptions and may lead to significant load on the browser tab process depending on topic update frequency.

The list of available services includes their request and response message format(s). It will later include descriptions and options to send service specific requests and see responses.

In future steps, the inspectors should add filtering options for topics like message format, etc. to support a fast and structured navigation of bigger systems.

5.10.3 Test Record Publishing

To quickly test subscription callbacks for which there is no active publisher yet, it is helpful to publish test data manually.
With this web interface, one can formulate *TopicDataRecords* in their respective JSON version and publish them as seen in Figure 26.

5.10.4 Graph Visualizer

The graph visualizer (Figure 27) is designed to give an overview over the nodes and their communication paths involved in a Ubi-Interact setup and more specif-

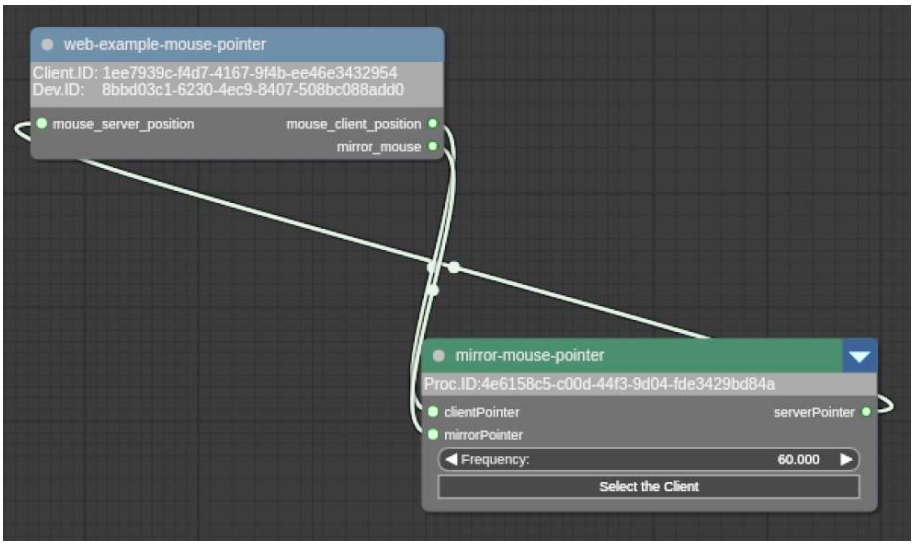


Figure 27: The example of mouse pointer mirroring (also accessible through the web frontend) visualized as a graph. Active publishing triggers an animation along respective edges.

ically a running *Session*.
It also serves to edit setups by e.g. graphically editing topic connections via drag-and-drop.

5.11 Applications

This section introduces some of the applications and environments that Ubi-Interact found use in. It may serve to illustrate how some of Ubi-Interact's concepts are applied in practice.

5.11.1 Image Processing Demonstrator

Intended to be both a test scenario for the handling of larger message sizes (images) and the integration with machine learning frameworks as well as a starting point for a useful extendable library of image processing functionality, a setup providing object recognition and optical character recognition (OCR) has been implemented together with Maximilian Schmidt[47].

[47] Schmidt, M. (2022). *Distributed Python Computation in Mixed Reality Environments* [Unpublished Master's Thesis]. TUM. `https://wiki.tum.de/display/infar/%5B22SS+-+MA%5D+Distributed+Python+Computation+in+Mixed+Reality+Environments`

In MR scenarios, it should serve as a library of image analysis modules that can be quickly integrated into a system including one or more cameras to make additional information available.
It was envisioned as a very dynamic and ad-hoc option for an individual participant to be able to take out their smartphone, open a provided web interface via a few clicks, place it into the environment or hold it onto a scene and have the added information available to them or the entire system depending on the use-case.
Of course, any permanently installed camera or other form of image source (maybe even realistic renderings of a virtual scene?) may be integrated just as well.

The setup consists of a web interface served through the web frontend that connects to an available camera stream and publishes images with a frequency of 10Hz. The same interface also opens a subscription for a topic of type *OBject2DList* that serves as a sink for extracted information coming from any *Processing Module* having analyzed the published image.
The frontend provides a GUI with a list of available *Processing Modules* that fit its own I/O schema of *Image* in, *Object2DList* out. Upon selecting a *Processing Module* and clicking start, it will create a new *Session* including the selected module mapped to its own topics.

One *Processing Module* was implemented for NodeJS. It imports an existing Tensorflow Single Shot MulitBox Detector (SSD) network[48] trained on the CoCo[164] dataset to recognize objects. Recognized objects are packed into the *Object2D* data model with name, image position and dimensions and consequently published as a list.

Another *Processing Module* allowing for OCR was implemented in Python by Maximilian Schmidt. It exists in three variants with different pre-processing steps,

5.11.2 Entertainment and Sports

A framework for super-human sports applications is being built by Eichhorn et al. at the FAR chair. Current application plans involve a quadcopter and teams of players wearing HMDs. All elements on the field need to communicate their game state and actions, potentially with immediate consequences to the game flow. [165]

This poses strict requirements on low-latency dissemination of messages and the potential to test out edge computation of game logic elements inside *Processing Modules*.

[48] `https://www.npmjs.com/package/@tensorflow-models/coco-ssd`

Figure 28: Results of the Python module for OCR on a test image.

5.11.3 Virtual Supermarket

A virtual supermarket scenario developed by Eichhorn et al. at the FAR chair has adopted Ubi-Interact for integration of a personal smartphone. [166]

The eventual goal is for the smartphone as an interaction device to be reflected as accurately and naturally in all its capabilities as possible.
To date, this involves relaying touch interaction, IMU data and potentially a camera stream to the VR application.

5.11.4 Serious Games

Some of the serious games developed by Plecher et al. at the FAR chair have integrated Ubi-Interact to provide multiplayer functionality. [167][168][169]

This is an excellent benchmark for Ubi-Interact's capabilities to deal with various requirements on number of connected clients, communication and synchronization speeds.

Developer support to implement application specific services and processing are also challenged by these developments.

5.11.5 Hololayer

Ubi-Interact was also integrated with the Hololayer system developed by Siemens Technology. [170]
This exemplifies a successful integration between industrial AR and research project. Concepts and events from both systems were used to trigger combined interactions and open up options for further experimentation, extensions and collaborations.

This integration also identified the aspects of security that are typically very low priority in research but essential in industrial settings and thus require increased priority for Ubi-Interact in order to deploy it conveniently in such environments.

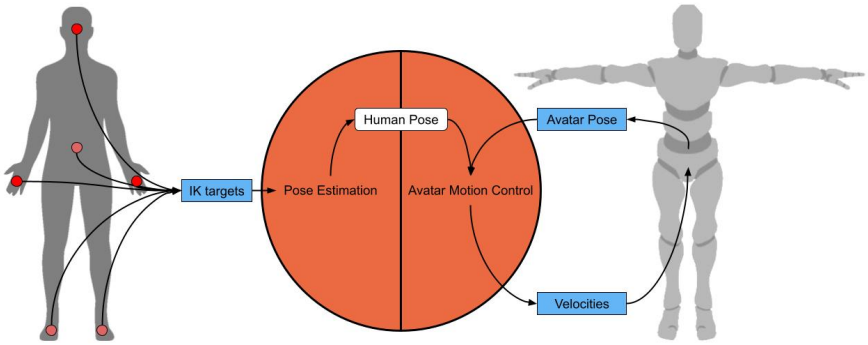


Figure 29: Three stages to the setup: 1) user tracking and VR visualization, 2) full body pose estimation and calculation of forces/velocities based on current avatar pose, 3) physically simulated avatar.

5.12 Modular Re-Embodiment with Ubi-Interact

The following setup was development during a bachelor's thesis[49] in cooperation with Leonard Goldstein to test the ability of Ubi-Interact to freely combine individual environments together in an interchangeable way. It takes on the same task of motor control over a physically simulated humanoid avatar as described in Section 3.5, albeit in a slightly different form.

The split between 1) an environment for VR visualization and human body tracking, 2) modules for the mapping between tracked body pose and applied forces to the avatar and 3) a physical simulation for the avatar in its environment was kept (see Figure 29).

The goal was then to build two alternatives for each step and demonstrate that Ubi-Interact can enable developers to freely connect any sequence of alternatives. The purpose was not to demonstrate one superior solution but to emphasize that solutions can be exchanged and compared easily through the integration with Ubi-Interact.

For one of the alternatives, the existing setup for Unity3D described in Section 3.5 was adopted and adjusted. The other alternative was implemented in a browser environment using Babylon.js.

For the motor control of the avatar, instead of applying forces for joint motors, it was decided to assume only the base functionality of any physics

[49] Goldstein, L. (2022). *Physical Embodiment in VR: Interchangeable Web-Based Modules using Ubi-Interact* [Unpublished Bachelor's Thesis]. TUM. `https://wiki.tum.de/display/infar/%5B22WS+-+BA%5D+Physical+Embodiment+in+VR%3A+Interchangeable+Web-Based+Modules+using+Ubi-Interact`

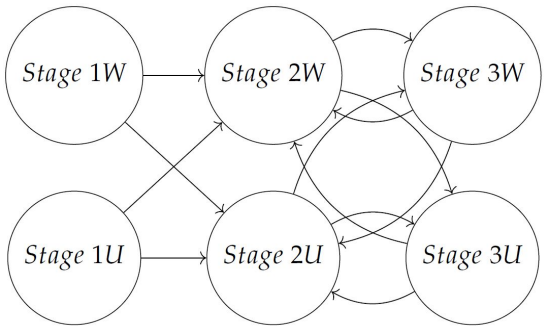


Figure 30: Interchangeable combination of stages. One alternative follows a web-based implementation (W), the other one is an adopted version of the Unity3D setup (U).

engine of applying linear and angular forces/velocities to bodies of mass.

5.12.1 Ubi-Interact setup

All three stages have been wrapped and described using the concepts of Ubi-Interact.

In stage 1, the element responsible for producing IK targets in stage 1 was wrapped in a *Device* with a single *Component* publishing data of format *Object3DList*[50] and carrying the tags "avatar", "user tracking", "ik", "targets", "ik targets" and "inverse kinematics". One Object3D includes position and orientation information reflecting one of the IK targets.

Stage 3 on the other hand provides a *Device* with two *Components*.
One *Component* would publish the current avatar body part poses - also of format *Object3DList* tagged with "avatar", "bones", "pose".
The other *Component* subscribed to topics - again of format *Object3DList*, carrying tags of "avatar", "bones", "control", "velocity", "linear", "angular" - awaiting velocities to be applied to individual body parts.

The data format of *Object3DList* fit for all three cases. It can reference elements via ID and name, i.e. specify IK target or body part from a string dictionary or reference auto-generated IDs of scene graph objects. Two times it reflects the actual 3-dimensional position and orientation description of an element, one time it is interpreted as linear and angular velocities to be applied. The way any stage and its data channels are identified is not based on their topic name or format, but by an overall *Device*, *Component* or *Processing Module*

[50] `https://github.com/SandroWeber/ubii-msg-formats/blob/develop/src/proto/dataStructure/object3d.proto`

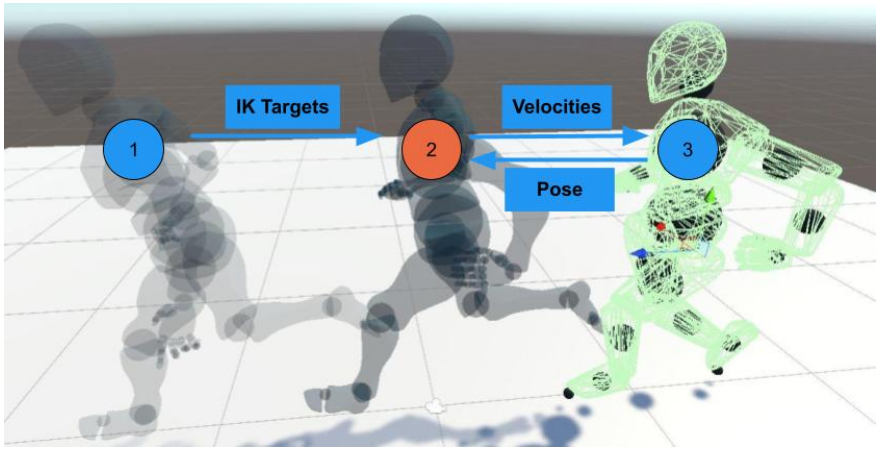


Figure 31: Side-by-side Unity3D visualization with consequent delay and deviation in pose: 1) user tracking and IK targets, 2) full body pose estimation and calculation of forces/velocities based on current avatar pose, 3) physically simulated avatar.

I/O profile including their tags and message formats. The topic is subsequently retrieved from an identified *Component*.

Stage 2 was represented in a *Processing Module*. Relying on the *Device* specifications of the other stages, it could then search for all *Components* required, link against their input/output channels and start its calculations once a setup has been established. Scaling this to multiple users is also possible, assuming a user (stage 1) first chooses or spawns their instance of an avatar (stage 3) and provides this ownership as input to stage 2. Stage 2 itself can either consist of a personal user instance of the required modules or one module acting for all users with additional input channels for user reference.

5.12.2 Unity3D

As mentioned before, the Unity3D implementation built upon the previously achieved setup for the NRP.
For convenience, all three stages were integrated into the same Unity3D scene but all communication went through the Ubi-Interact master node.
Individual components were then wrapped into *Components*, *Devices* and *Processing Modules* according to Ubi-Interact specifications. Stage 2 specifically as a *Processing Module* was run in "free" mode since its calculations happened within the Unity3D rendering cycle and thus must happen outside of any iterations controlled by Ubi-Interact.

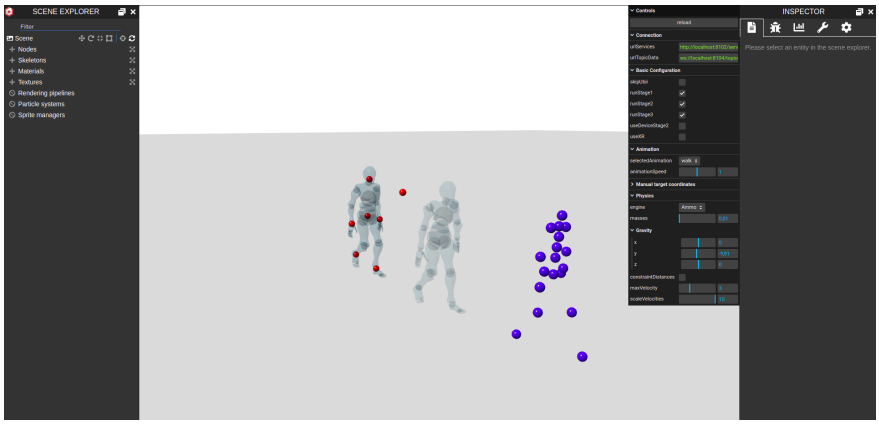


Figure 32: Side-by-side BabylonJS visualization with debugging GUI. From left to right: IK targets (red), processing and simulated body (here only visualized as spheres).

5.12.3 Web

Much like in the case of the Unity3D application, all stages had to be communicated with Ubi-Interact. Implementation was accomplished by Leonard Goldstein.

Especially the body pose estimation from IK targets proved to be challenging, as the choice of existing IK libraries involving constraints of the human body are limited. The solutions of the JS version of stage 2 are therefore in general inferior to the Unity3D equivalent.

5.12.4 Results

Expressing the functionality of the different stages via Ubi-Interact concepts proved helpful with their discoverability and interchangeability. It made it easy for both alternatives to provide the same profile of functionality (while potentially differing in their implementation) and discover the other stages related to them. Either alternative could be run and would not require any manual reconfiguration of the overall setup.

It also allowed for a direct comparison of stage 2 implementations. Either option can be exchanged or even run in parallel and the results be compared side-by-side during runtime.

As indicated in Figure 29, stage 2 actually consists of two separate steps. In the setup described here, they were part of the same *Processing Module*. If

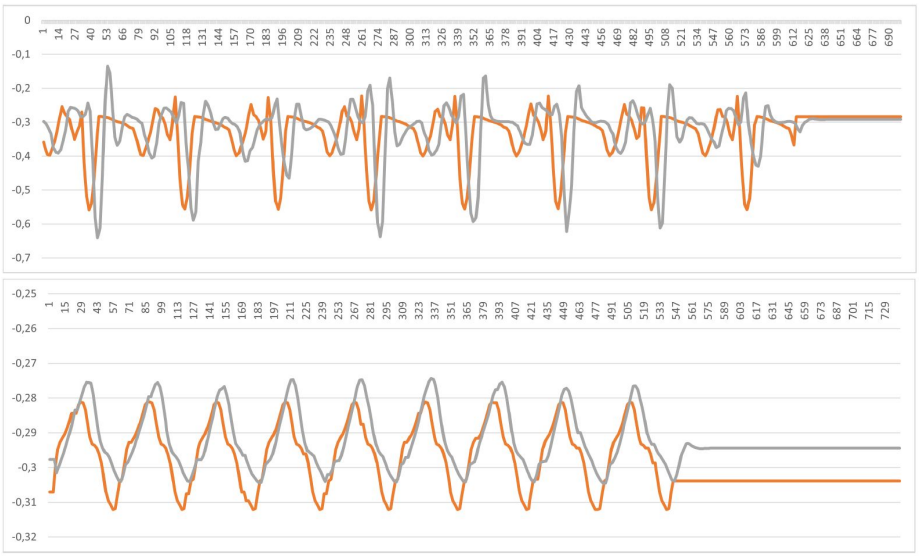


Figure 33: Evaluation of latency (x-axis, milliseconds) and displacement in X-coordinates (y-axis, meters) between visual output of stage 1 and stage 3. The upper graph shows the Unity3D case, the lower graph shows the web application. Adopted from Leonard Goldstein's thesis.

in the future the setup was to include full body tracking for the user, stage 2 would be split into two separate modules. The first module would then get instantiated depending on what data stage 1 can provide. This reflects the goal for Ubi-Interact to conditionally and contextually decide what parts of an implementation to rely on.
To avoid extra network latency and bandwidth use, both modules should ideally still run on the same *Node*. The *Node* can thus act as an event hub to the rest of the process without having to go through the broker, but keep the option for external input/output to/from the module from outside the *Node*.

Detailed measurements on latency and displacement between solutions are available in Leonard Goldstein's thesis mentioned above. End-to-end latency between movement and visual feedback was well within the limits mentioned in Section 3.1.4 for the particular test environment (Figure 33).

6 Future Work

Naturally, there is a continuous list of improvements to be made - whether it be for security, performance, convenience or completely new features.
The following sections list some of the items on top of the list.

6.1 Node features

New Ubi-Interact nodes are typically implemented whenever they are needed for the targeted environment. Their development and feature list extension is certainly a community effort.
Sometimes, not all features are required at the time of implementation. Features above the base communication, too, are added whenever there is demand.

The use of Protocol Buffers schema definitions for all messages means that nodes usually do not degrade over time because of changes to the master node, only that they may be missing some additional features.

It is a continuous effort to keep nodes and their API consistent and up-to-date.

6.2 Pub/sub broker performance

For the broker itself, one of the main performance gains would certainly be to move from topics being handled as integers instead of strings.
This requires implementation of a hashing function consistent for all nodes, after which lookup and comparison speed would certainly be increased.

All plans to improve broker performance, however, have to be weighted against the option to adopt an existing pub/sub system if possible.
This decision will be postponed until the additional features of Ubi-Interact for conditional notifications have been tested, their useful has been (dis)proven and an evaluation has been made on the possibility to integrate such features into an existing code base.

6.3 Time Synchronization

Although *TopicDataRecords* include the possibility for timestamps, there is currently no time synchronization feature between nodes.

One possible solution to this could be for the master node to send configu-

ration for a shared Network Time Protocol[51][171] (NTP) server during initial registration of a client.
Alternatively, if all nodes run in the same local area network, the Precision Time Protocol[52][172] (PTP) should produce higher precision.

6.4 Security

When it comes to support for security and privacy features, there is certainly a list of additional features that need to be implemented in Ubi-Interact.

The most general and already available method of securing a Ubi-Interact deployment is on the level of a web server proxy or request access.
Nginx[53] is one of the most prominent tools for web server configuration and there are several options to include user authentication and access control inside an Nginx configuration for any request before forwarding it to Ubi-Interact.

Beyond this broad level of security, Ubi-Interact should offer services to integrate with existing user and access management solutions.
This could then be leveraged in *Sessions* as an additional client and scope control for any topic and service data, forming communication groups with restricted access that are isolated against each other.
Such a mechanism would entail additional *Session* login services and checks that need to be specified by the creating node.

On the lowest level of integration, this authentication / authorization service should be made available within *NotifyConditions* to allow checks on a per-topic and per-notification basis.

6.5 Conditional Publish/Subscribe

The concepts introduced in Section 5.7.2 for *NotifyConditions* need to be fully evaluated.

This will require bigger in-situ studies to determine their usefulness for different scenarios of efficiency, security and privacy.
How Ubi-Interact can support in managing the added complexity and decision making is another aspect to investigate.

Equally, their impact on performance of the broker and how to implement them efficiently (e.g. conditions being shared and only being evaluated once for

[51] `http://www.ntp.org/`
[52] `https://endruntechnologies.com/pdf/PTP-1588.pdf`
[53] `https://nginx.org`

updates on the values they depend on, non-critical conditions being evaluated in tolerable time intervals instead of triggering with each update, etc.) is something that must be carefully balanced.

6.6 Intent

Llorens-Bonilla et al.[83] already mentioned how crucial it is for their robotic limbs assistance system to understand its user's intent.
For any technological extension to human agency, cooperative task performance is a crucial factor to investigate.

How to generally represent agent intent in a system like Ubi-Interact and make it a shared context and resource to build upon for all elements involved in an agent's activity is another point of future research.
This does not only involve human actions in physical scenarios, but also more abstract goals and interactions in virtual spaces.

With this, modules working in a tight loop with user input may be able to e.g. compensate momentary latency spikes and bridge gaps of lost agency.
Alternatively they could assist in or completely take over subtasks more effectively.

To give some possible direcions, methods of human activity recognition[173] or eye-tracking hardware integration as a *Component* could surely contribute in this regard.

7 Conclusion

In conclusion, it is the hope of the author that this work as a framework can contribute to flexible and re-usable setups.

Ubi-Interact is a framework designed to facilitate ambiguous Ubiquitous Mixed Reality environments where agents are a priori unfamiliar with parts of their environment and being part of the system does not imply being eligible to receive all information at all times.

If Ubi-Interact allows for setups to be more maintainable over time and experimentation to happen more rapidly, then it has fulfilled its purpose.
It should help keep experiments alive, functional and quickly adjustable to new circumstances at a later point in time.
If different approaches need to be evaluated against each other, the functionality provided should ease the process as well.

Ideally, this fosters cooperation by reducing the effort to combine setups developed in parallel.
Reproducing and testing approaches in-situ in combination with other systems working in the same environment should not impose impeding effort in time and complexity handling.

Lastly, it is the hope that users of Ubi-Interact find its concepts convenient to use and beneficial in their application. Ubi-Interact must be continuously improved with the help and feedback of its users in order to advance.

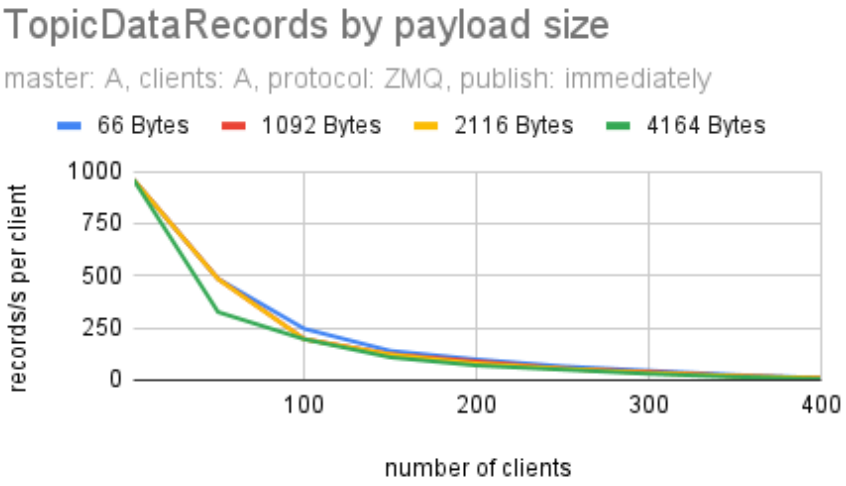
List of Figures

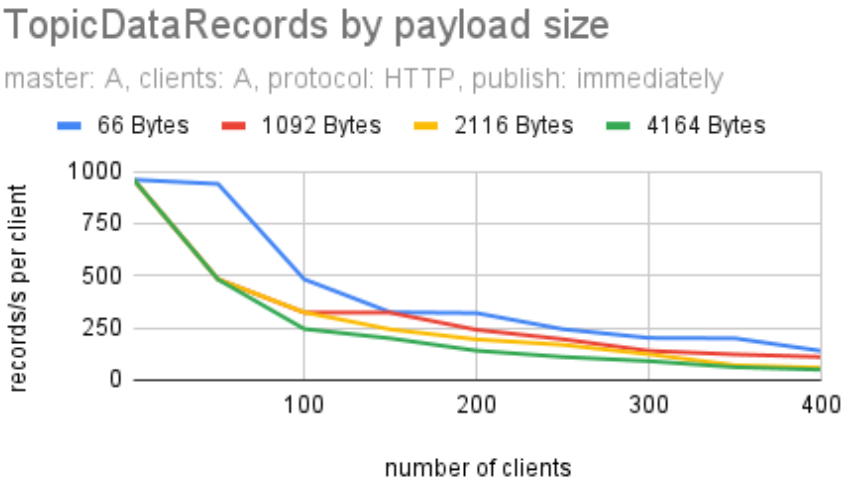
List of Tables

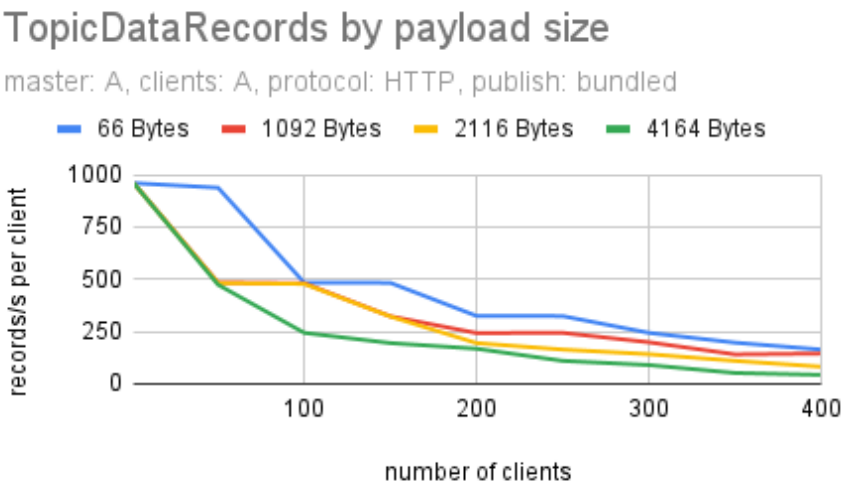
A Appendix

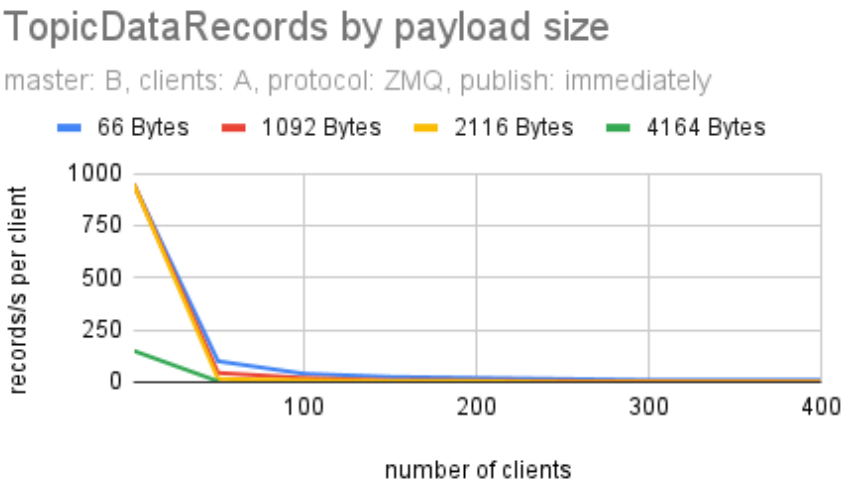
A.1 Ubi-Interact broker performance

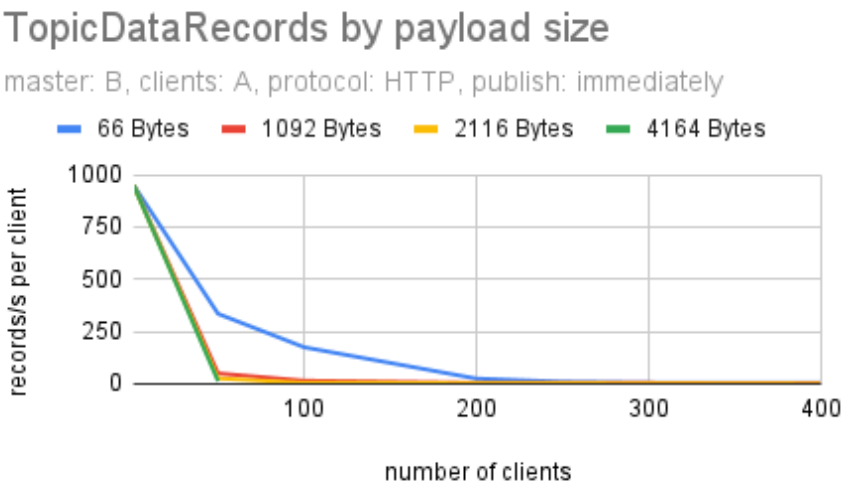
For reference - hardware abbreviations:
A) AMD Ryzen 5 3600 6-Core Processor (3.60 GHz) CPU, 16GB RAM under Ubuntu 20.04.5 LTS
B) Raspberry Pi 4 Model B under Raspbian 11

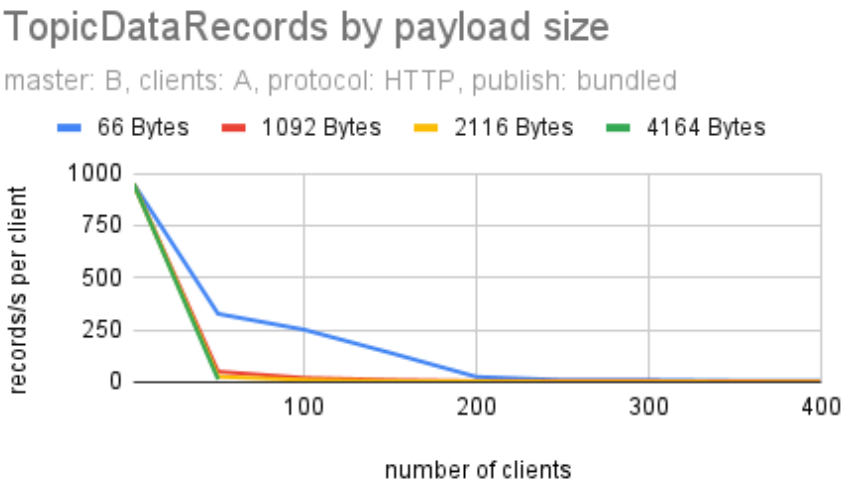


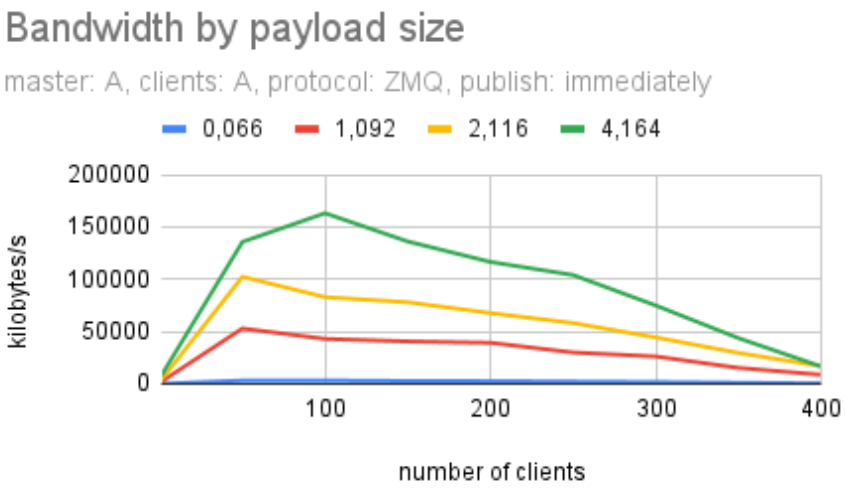


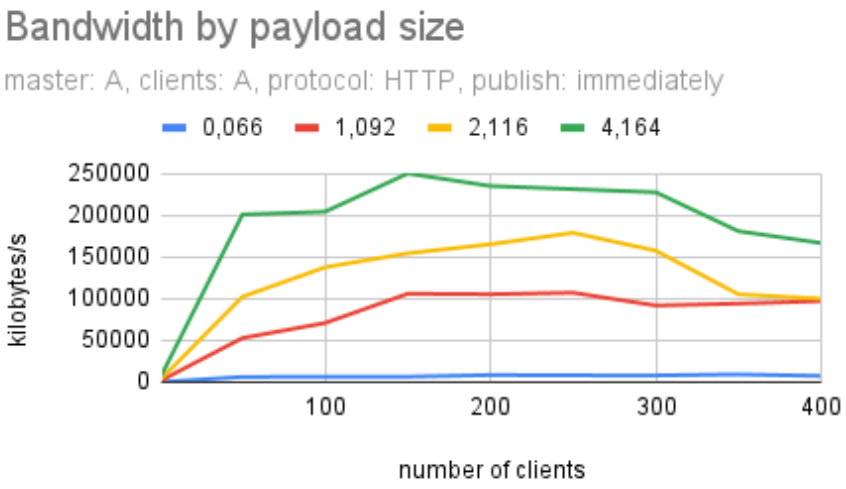

TopicDataRecords by payload size
master: A, clients: A, protocol: HTTP, publish: bundled
66 Bytes
1092 Bytes
2116 Bytes
4164 Bytes
1000
750
500
250
0
records/s per client
100
200
300
400
number of clients

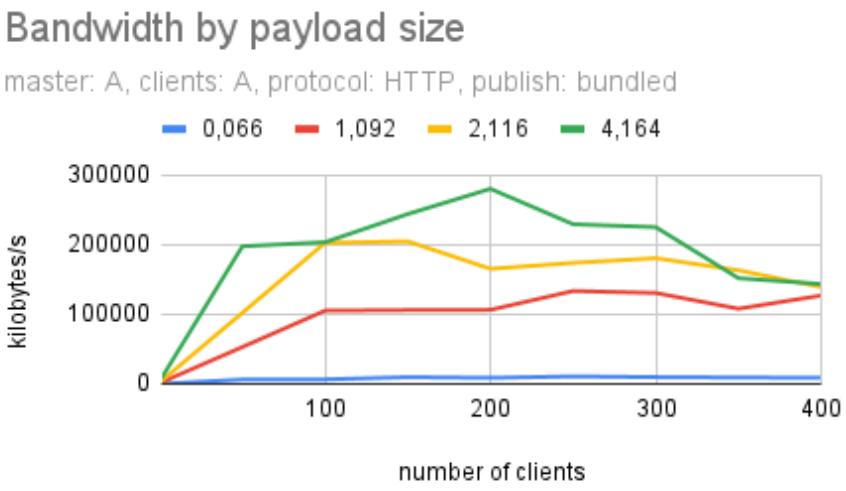

TopicDataRecords by payload size
master: B, clients: A, protocol: ZMQ, publish: immediately
66 Bytes
1092 Bytes
2116 Bytes
4164 Bytes
1000
750
500
250
0
records/s per client
100
200
300
400
number of clients

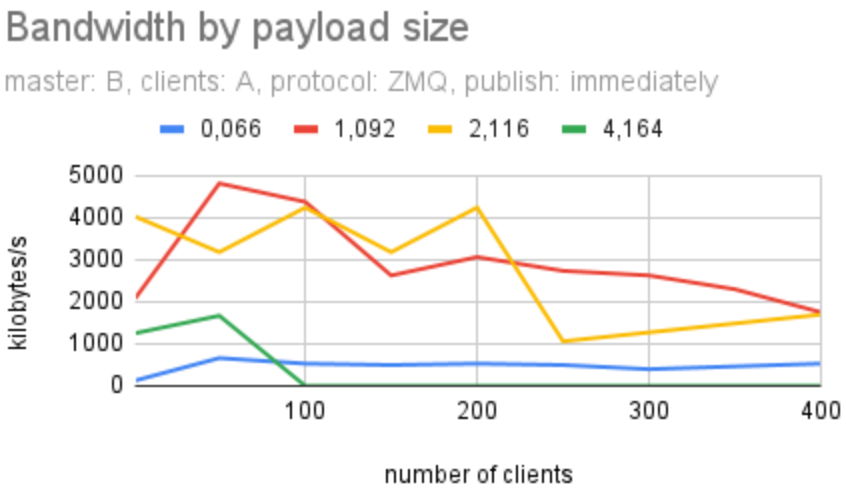

TopicDataRecords by payload size
master: B, clients: A, protocol: HTTP, publish: immediately
66 Bytes
1092 Bytes
2116 Bytes
4164 Bytes
1000
750
500
250
0
records/s per client
100
200
300
400
number of clients

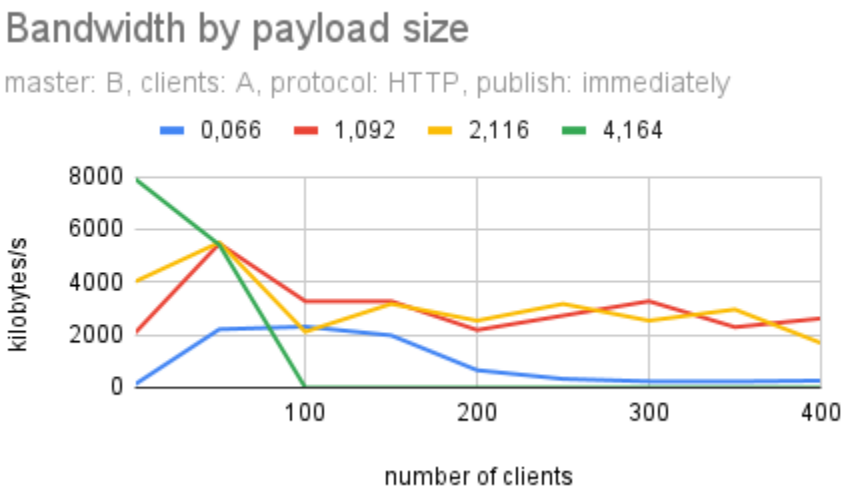

TopicDataRecords by payload size
master: B, clients: A, protocol: HTTP, publish: bundled
66 Bytes
1092 Bytes
2116 Bytes
4164 Bytes
1000
750
500
250
0
records/s per client
100
200
300
400
number of clients

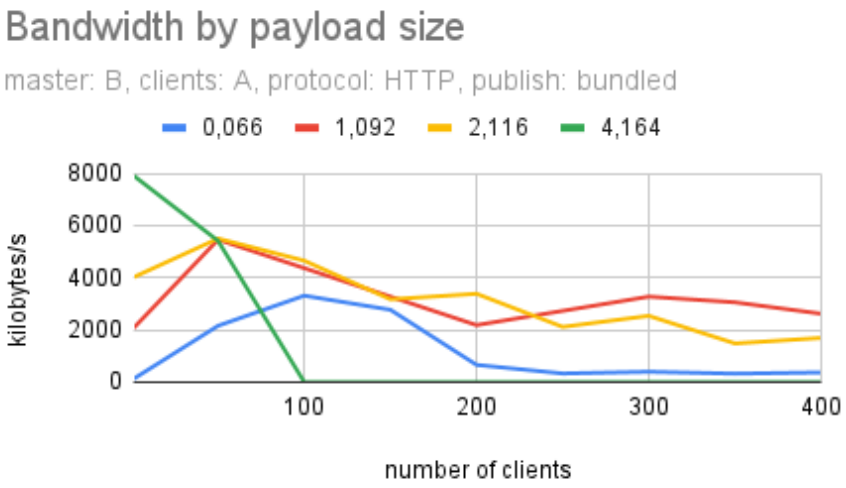

Bandwidth by payload size
master: A, clients: A, protocol: ZMQ, publish: immediately
0,066
1,092
2,116
4,164
200000
150000
100000
50000
0
kilobytes/s
100
200
300
400
number of clients


Bandwidth by payload size
master: A, clients: A, protocol: HTTP, publish: immediately
0,066
1,092
2,116
4,164
250000
200000
150000
100000
50000
0
kilobytes/s
100
200
300
400
number of clients


Bandwidth by payload size
master: A, clients: A, protocol: HTTP, publish: bundled
0,066
1,092
2,116
4,164
300000
200000
100000
0
kilobytes/s
100
200
300
400
number of clients


Bandwidth by payload size
master: B, clients: A, protocol: ZMQ, publish: immediately
0,066
1,092
2,116
4,164
5000
4000
3000
2000
1000
0
kilobytes/s
100
200
300
400
number of clients


Bandwidth by payload size
master: B, clients: A, protocol: HTTP, publish: immediately
0,066
1,092
2,116
4,164
8000
6000
4000
2000
0
kilobytes/s
100
200
300
400
number of clients


Bandwidth by payload size
master: B, clients: A, protocol: HTTP, publish: bundled
0,066
1,092
2,116
4,164
8000
6000
4000
2000
0
kilobytes/s
100
200
300
400
number of clients

A.2 Acronyms

PR	Physical Reality
VR	Virtual Reality
AR	Augmented Reality
AV	Augmented Virtuality
MR	Mixed Reality
UMR	Ubiquitous Mixed Reality
PE	Physical Environment
VE	Virtual Environment
ME	Mixed Environment
IMU	Internal Measurement Unit
IoT	Internet of Things
HCI	Human-Computer-Interaction
HRI	Human-Robot-Interaction
NRP	Neurorobotics Platform
IVA	Intelligent Virtual Agent / Assistant

References

[1] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

[2] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6):1309–1332, 2016.

[3] William A McNeely. Robotic graphics: a new approach to force feedback for virtual reality. In *Proceedings of IEEE Virtual Reality Annual International Symposium*, pages 336–341. IEEE, 1993.

[4] Parastoo Abtahi, Benoit Landry, Jackie Yang, Marco Pavone, Sean Follmer, and James A Landay. Beyond the force: Using quadcopters to appropriate objects and the environment for haptics in virtual reality. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2019.

[5] Antonio Frisoli, Fabio Salsedo, Massimo Bergamasco, Bruno Rossi, and Maria C Carboncini. A force-feedback exoskeleton for upper-limb rehabilitation in virtual reality. *Applied Bionics and Biomechanics*, 6(2):115–126, 2009.

[6] Paul Milgram and Fumio Kishino. A taxonomy of mixed reality visual displays. *IEICE TRANSACTIONS on Information and Systems*, 77(12):1321–1329, 1994.

[7] Mark EJ Newman. Complex systems: A survey. *arXiv preprint arXiv:1112.1440*, 2011.

[8] Maximilian Speicher, Brian D Hall, and Michael Nebeling. What is mixed reality? In *Proceedings of the 2019 CHI conference on human factors in computing systems*, pages 1–15, 2019.

[9] Heinz Von Foerster. On constructing a reality. In *Environmental design research*, pages 35–46. Routledge, 2018.

[10] Nick Bostrom. Are we living in a computer simulation? *The philosophical quarterly*, 53(211):243–255, 2003.

[11] Robert J Bradbury et al. Matrioshka brains. *preprint at http://www. aeiveos. com/˜ bradbury/MatrioshkaBrains/MatrioshkaBrains. html*, 2001.

[12] James J Cummings and Jeremy N Bailenson. How immersive is enough? a meta-analysis of the effect of immersive technology on user presence. *Media psychology*, 19(2):272–309, 2016.

[13] Ayoung Suh and Jane Prophet. The state of immersive technology research: A literature analysis. *Computers in Human Behavior*, 86:77–90, 2018.

[14] Michael Wooldridge. Agent-based software engineering. *IEE Proceedings-software*, 144(1):26–37, 1997.

[15] Carolina Cruz-Neira, Daniel J Sandin, and Thomas A DeFanti. Surround-screen projection-based virtual reality: the design and implementation of the cave. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 135–142, 1993.

[16] Cagatay Demiralp, Cullen D Jackson, David B Karelitz, Song Zhang, and David H Laidlaw. Cave and fishtank virtual-reality displays: A qualitative and quantitative comparison. *IEEE transactions on visualization and computer graphics*, 12(3):323–330, 2006.

[17] Marc Aurel Schnabel, Xiangyu Wang, Hartmut Seichter, and Tom Kvan. From virtuality to reality and back. 2007.

[18] Peter W Halligan, Gereon R Fink, John C Marshall, and Giuseppe Vallar. Spatial cognition: evidence from visual neglect. *Trends in cognitive sciences*, 7(3):125–133, 2003.

[19] Steve Mann, Tom Furness, Yu Yuan, Jay Iorio, and Zixin Wang. All reality: Virtual, augmented, mixed (x), mediated (x, y), and multimediated reality. *arXiv preprint arXiv:1804.08386*, 2018.

[20] Mark Weiser. The computer for the 21 st century. *Scientific american*, 265(3):94–105, 1991.

[21] Mark Weiser. Hot topics-ubiquitous computing. *Computer*, 26(10):71–72, 1993.

[22] Mark Weiser and John Seely Brown. Designing calm technology. *PowerGrid Journal*, 1(1):75–85, 1996.

[23] Mark Weiser and John Seely Brown. The coming age of calm technology. In *Beyond calculation*, pages 75–85. Springer, 1997.

[24] Donald A Norman. *The invisible computer: why good products can fail, the personal computer is so complex, and information appliances are the solution*. 1998.

[25] Kalle Lyytinen and Youngjin Yoo. Ubiquitous computing. *Communications of the ACM*, 45(12):63–96, 2002.

[26] Michael Friedewald and Oliver Raabe. Ubiquitous computing: An overview of technology impacts. *Telematics and Informatics*, 28(2):55–65, 2011.

[27] Johann Bizer, Kai Dingel, Benjamin Fabian, Oliver Günther, Markus Hansen, Michael Klafft, Jan Möller, and Sarah Spiekermann. *Technikfolgenabschätzung: Ubiquitäres Computing und Informationelle Selbstbestimmung Studie im Auftrag des Bundesministeriums für Bildung und Forschung ; TAUCIS*. 2006.

[28] Guruduth Banavar and Abraham Bernstein. Software infrastructure and design challenges for ubiquitous computing applications. *Communications of the ACM*, 45(12):92–96, 2002.

[29] Felix Ocker, Birgit Vogel-Heuser, and Christiaan JJ Paredis. A framework for merging ontologies in the context of smart factories. *Computers in Industry*, 135:103571, 2022.

[30] Mahadev Satyanarayanan. Pervasive computing: Vision and challenges. *IEEE Personal communications*, 8(4):10–17, 2001.

[31] Albrecht Schmidt. *Ubiquitous computing-computing in context*. Lancaster University (United Kingdom), 2003.

[32] Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *1994 first workshop on mobile computing systems and applications*, pages 85–90. IEEE, 1994.

[33] Nathalie Bricon-Souf and Conrad R Newman. Context awareness in health care: A review. *international journal of medical informatics*, 76(1):2–12, 2007.

[34] Ola Henfridsson and Rikard Lindgren. Multi-contextuality in ubiquitous computing: Investigating the car case through action research. *Information and organization*, 15(2):95–124, 2005.

[35] Davide Figo, Pedro C Diniz, Diogo R Ferreira, and Joao MP Cardoso. Preprocessing techniques for context recognition from accelerometer data. *Personal and Ubiquitous Computing*, 14(7):645–662, 2010.

[36] Louis Atallah and Guang-Zhong Yang. The use of pervasive sensing for behaviour profiling—a survey. *Pervasive and mobile computing*, 5(5):447–464, 2009.

[37] Gwo-Dong Chen, Chih-Kai Chang, and Chih-Yeh Wang. Ubiquitous learning website: Scaffold learners by mobile devices with information-aware techniques. *Computers & education*, 50(1):77–90, 2008.

[38] Hariharasudhan Viswanathan, Baozhi Chen, and Dario Pompili. Research challenges in computation, communication, and context awareness for ubiquitous healthcare. *IEEE Communications Magazine*, 50(5):92–99, 2012.

[39] Vicki Jones and Jun H Jo. Ubiquitous learning environment: An adaptive teaching system using ubiquitous technology. In *Beyond the comfort zone: Proceedings of the 21st ASCILITE Conference*, volume 468, page 474. Perth, Western Australia, 2004.

[40] Adam Greenfield. *Everyware: The dawning age of ubiquitous computing*. New Riders, 2010.

[41] Hany F Atlam and Gary B Wills. Iot security, privacy, safety and ethics. In *Digital twin technologies and smart cities*, pages 123–149. Springer, 2020.

[42] Thomas R Gruber. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993.

[43] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific american*, 284(5):34–43, 2001.

[44] Nigel Shadbolt, Tim Berners-Lee, and Wendy Hall. The semantic web revisited. *IEEE intelligent systems*, 21(3):96–101, 2006.

[45] Deborah L McGuinness, Frank Van Harmelen, et al. Owl web ontology language overview. *W3C recommendation*, 10(10):2004, 2004.

[46] Nuno Pereira, Anthony Rowe, Michael W Farb, Ivan Liang, Edward Lu, and Eric Riebling. Arena: The augmented reality edge networking architecture. In *2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 479–488. IEEE, 2021.

[47] Hannu Karvonen, Tuomo Kujala, and Pertti Saariluoma. In-car ubiquitous computing: driver tutoring messages presented on a head-up display. In *2006 IEEE Intelligent Transportation Systems Conference*, pages 560–565. IEEE, 2006.

[48] Reiner Jedermann and Walter Lang. The benefits of embedded intelligence–tasks and applications for ubiquitous computing in logistics. In *The internet of things*, pages 105–122. Springer, 2008.

[49] Kwanho Kim, Hyunjin Kim, Sang-Kuk Kim, and Jae-Yoon Jung. i-rm: An intelligent risk management framework for context-aware ubiquitous cold chain logistics. *Expert Systems with Applications*, 46:463–473, 2016.

[50] W Keith Edwards and Rebecca E Grinter. At home with ubiquitous computing: Seven challenges. In *International conference on ubiquitous computing*, pages 256–272. Springer, 2001.

[51] Tatsuya Yamazaki. Beyond the smart home. In *2006 International Conference on Hybrid Information Technology*, volume 2, pages 350–355. IEEE, 2006.

[52] Marianthi Theoharidou, Nikos Tsalis, and Dimitris Gritzalis. Smart home solutions for healthcare: privacy in ubiquitous computing infrastructures. *Handbook of smart homes, health care and well-being*, 2014.

[53] Ken Sakamura and Noboru Koshizuka. Ubiquitous computing technologies for ubiquitous learning. In *IEEE International Workshop on Wireless and Mobile Technologies in Education (WMTE'05)*, pages 11–20. IEEE, 2005.

[54] Yong-Woon Moon, Hae-Sun Jung, and Chang-Sung Jeong. Context-awareness in battlefield using ubiquitous computing: Network centric warfare. In *2010 10th IEEE International Conference on Computer and Information Technology*, pages 2873–2877. IEEE, 2010.

[55] Konstantina Kilteni, Ilias Bergstrom, and Mel Slater. Drumming in immersive virtual reality: the body shapes the way we play. *IEEE transactions on visualization and computer graphics*, 19(4):597–605, 2013.

[56] Sang In Jung, Na Kyung Lee, Kyung Woo Kang, Kyoung Kim, and Youn Lee Do. The effect of smartphone usage time on posture and respiratory function. *Journal of physical therapy science*, 28(1):186–189, 2016.

[57] Angelo Maravita and Atsushi Iriki. Tools for the body (schema). *Trends in cognitive sciences*, 8(2):79–86, 2004.

[58] Andy Clark and David Chalmers. The extended mind. *analysis*, 58(1):7–19, 1998.

[59] Konstantina Kilteni, Jean-Marie Normand, Maria V Sanchez-Vives, and Mel Slater. Extending body space in immersive virtual reality: a very long arm illusion. *PloS one*, 7(7):e40867, 2012.

[60] Meel Velliste, Sagi Perel, M Chance Spalding, Andrew S Whitford, and Andrew B Schwartz. Cortical control of a prosthetic arm for self-feeding. *Nature*, 453(7198):1098–1101, 2008.

[61] Maurice Merleau-Ponty. *Phenomenology of perception*. Routledge, 2013.

[62] Helena De Preester. Technology and the body: the (im) possibilities of re-embodiment. *Foundations of science*, 16(2):119–137, 2011.

[63] Melita J Giummarra, Stephen J Gibson, Nellie Georgiou-Karistianis, and John L Bradshaw. Mechanisms underlying embodiment, disembodiment and loss of embodiment. *Neuroscience & Biobehavioral Reviews*, 32(1):143–160, 2008.

[64] John Schwoebel and H Branch Coslett. Evidence for multiple, distinct representations of the human body. *Journal of cognitive neuroscience*, 17(4):543–553, 2005.

[65] Shaun Gallagher and Jonathan Cole. Body image and body schema in a deafferented subject. *The journal of mind and behavior*, pages 369–389, 1995.

[66] Federica Buongiorno. Embodiment, disembodiment and re-embodiment in the construction of the digital self. *HUMANA. MENTE Journal of Philosophical Studies*, 12(36):310–330, 2019.

[67] Konstantina Kilteni, Raphaela Groten, and Mel Slater. The sense of embodiment in virtual reality. *Presence: Teleoperators and Virtual Environments*, 21(4):373–387, 2012.

[68] Jane Aspell, Bigna Lenggenhager, and Olaf Blanke. Multisensory perception and bodily self-consciousness: from out-of body to inside-body experience. *The neural bases of multisensory processes*, (BOOK_CHAP), 2011.

[69] Bigna Lenggenhager, Michael Mouthon, and Olaf Blanke. Spatial aspects of bodily self-consciousness. *Consciousness and cognition*, 18(1):110–117, 2009.

[70] Shaun Gallagher. Philosophical conceptions of the self: implications for cognitive science. *Trends in cognitive sciences*, 4(1):14–21, 2000.

[71] Manos Tsakiris, Gita Prabhu, and Patrick Haggard. Having a body versus moving your body: How agency structures body-ownership. *Consciousness and cognition*, 15(2):423–432, 2006.

[72] Olaf Blanke and Thomas Metzinger. Full-body illusions and minimal phenomenal selfhood. *Trends in cognitive sciences*, 13(1):7–13, 2009.

[73] Luna Dolezal. The remote body: The phenomenology of telepresence and re-embodiment. *Human Technology: An Interdisciplinary Journal on Humans in ICT Environments*, 2009.

[74] Manos Tsakiris, Simone Schütz-Bosbach, and Shaun Gallagher. On agency and body-ownership: Phenomenological and neurocognitive reflections. *Consciousness and cognition*, 16(3):645–660, 2007.

[75] Polona Caserman, Michelle Martinussen, and Stefan Göbel. Effects of end-to-end latency on user experience and performance in immersive virtual reality applications. In *Joint International Conference on Entertainment Computing and Serious Games*, pages 57–69. Springer, 2019.

[76] Thomas Waltemate, Irene Senna, Felix Hülsmann, Marieke Rohde, Stefan Kopp, Marc Ernst, and Mario Botsch. The impact of latency on perceptual judgments and motor performance in closed-loop interaction in virtual reality. In *Proceedings of the 22nd ACM conference on virtual reality software and technology*, pages 27–35, 2016.

[77] Daniel Roth and Marc Erich Latoschik. Construction of a validated virtual embodiment questionnaire. *arXiv preprint arXiv:1911.10176*, 2019.

[78] Valeria I Petkova, Mehrnoush Khoshnevis, and H Henrik Ehrsson. The perspective matters! multisensory integration in ego-centric reference frames determines full-body ownership. *Frontiers in psychology*, 2:35, 2011.

[79] Geoffrey Gorisse, Olivier Christmann, Etienne Armand Amato, and Simon Richir. First-and third-person perspectives in immersive virtual environments: presence and performance analysis of embodied users. *Frontiers in Robotics and AI*, 4:33, 2017.

[80] Henrique Galvan Debarba, Sidney Bovet, Roy Salomon, Olaf Blanke, Bruno Herbelin, and Ronan Boulic. Characterizing first and third person viewpoints and their alternation for embodied interaction in virtual reality. *PloS one*, 12(12):e0190109, 2017.

[81] Shunichi Kasahara, Keina Konno, Richi Owaki, Tsubasa Nishi, Akiko Takeshita, Takayuki Ito, Shoko Kasuga, and Junichi Ushiba. Malleable embodiment: changing sense of embodiment by spatial-temporal deformation of virtual human body. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 6438–6448, 2017.

[82] Nick Yee, Nicolas Ducheneaut, and Jason Ellis. The tyranny of embodiment. *Artifact: Journal of Design Practice*, 2(2):88–93, 2008.

[83] Baldin Llorens Bonilla and H Harry Asada. A robot on the shoulder: Coordinated human-wearable robot control using coloured petri nets and partial least squares predictions. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 119–125. IEEE, 2014.

[84] Christian I Penaloza and Shuichi Nishio. Bmi control of a third arm for multitasking. *Science Robotics*, 3(20):eaat1228, 2018.

[85] Nikolas Martens, Robert Jenke, Mohammad Abu-Alqumsan, Christoph Kapeller, Christoph Hintermüller, Christoph Guger, Angelika Peer, and Martin Buss. Towards robotic re-embodiment using a brain-and-body-computer interface. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5131–5132. IEEE, 2012.

[86] Ricardo Chavarriaga and José del R Millán. Learning from eeg error-related potentials in noninvasive brain-computer interfaces. *IEEE transactions on neural systems and rehabilitation engineering*, 18(4):381–388, 2010.

[87] Christopher Stanton, Anton Bogdanovych, and Edward Ratanasena. Teleoperation of a humanoid robot using full-body motion capture, example movements, and machine learning. In *Proc. Australasian Conference on Robotics and Automation*, volume 8, page 51, 2012.

[88] Aravind Sivakumar, Kenneth Shaw, and Deepak Pathak. Robotic telekinesis: learning a robotic hand imitator by watching humans on youtube. *arXiv preprint arXiv:2202.10448*, 2022.

[89] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2149–2154. IEEE, 2004.

[90] Trevor Bekolay, James Bergstra, Eric Hunsberger, Travis DeWolf, Terrence C Stewart, Daniel Rasmussen, Xuan Choo, Aaron Voelker, and Chris Eliasmith. Nengo: a python tool for building large-scale functional brain models. *Frontiers in neuroinformatics*, 7:48, 2014.

[91] Egidio Falotico, Lorenzo Vannucci, Alessandro Ambrosano, Ugo Albanese, Stefan Ulbrich, Juan Camilo Vasquez Tieck, Georg Hinkel, Jacques Kaiser, Igor Peric, Oliver Denninger, et al. Connecting artificial brains to robots in a comprehensive simulation framework: the neurorobotics platform. *Frontiers in neurorobotics*, 11:2, 2017.

[92] Marc-Oliver Gewaltig and Markus Diesmann. Nest (neural simulation tool). *Scholarpedia*, 2(4):1430, 2007.

[93] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. {TensorFlow}: a system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016.

[94] Michael A Johnson and Mohammad H Moradi. *PID control*. Springer, 2005.

[95] J Camilo Vasquez Tieck, Sandro Weber, Terrence C Stewart, Arne Roennau, and Rüdiger Dillmann. Triggering robot hand reflexes with human emg data using spiking neurons. In *International Conference on Intelligent Autonomous Systems*, pages 902–916. Springer, 2018.

[96] Sandro Weber, Linda Rudolph, Christian Eichhorn, Daniel Dyrda, David A Plecher, Gudrun Klinker, et al. Frameworks enabling ubiquitous mixed reality applications across dynamically adaptable device configurations. *Frontiers in Virtual Reality*, page 36, 2022.

[97] Antonella Maselli and Mel Slater. The building blocks of the full body ownership illusion. *Frontiers in human neuroscience*, 7:83, 2013.

[98] Deepak Tolani, Ambarish Goswami, and Norman I Badler. Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical models*, 62(5):353–388, 2000.

[99] Richard M Murray, Zexiang Li, and S Shankar Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 2017.

[100] Robert J Griffin, Georg Wiedebach, Sylvain Bertrand, Alexander Leonessa, and Jerry Pratt. Walking stabilization using step timing and location adjustment on the humanoid robot, atlas. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 667–673. IEEE, 2017.

[101] Ken Birman and Thomas Joseph. Exploiting virtual synchrony in distributed systems. In *Proceedings of the eleventh ACM Symposium on Operating systems principles*, pages 123–138, 1987.

[102] Patrick Th Eugster, Pascal A Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM computing surveys (CSUR)*, 35(2):114–131, 2003.

[103] Joao Pereira, Francoise Fabret, Francois Llirbat, and Dennis Shasha. Efficient matching for web-based publish/subscribe systems. In *International conference on cooperative information systems*, pages 162–173. Springer, 2000.

[104] Alexis Campailla, Sagar Chaki, Edmund Clarke, Somesh Jha, and Helmut Veith. Efficient filtering in publish-subscribe systems using binary decision diagrams. In *Proceedings of the 23rd International Conference on Software Engineering. ICSE 2001*, pages 443–452. IEEE, 2001.

[105] Jay Kreps, Neha Narkhede, Jun Rao, et al. Kafka: A distributed messaging system for log processing. In *Proceedings of the NetDB*, volume 11, pages 1–7, 2011.

[106] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[107] Morgan Quigley, Brian Gerkey, and William D Smart. *Programming Robots with ROS: a practical introduction to the Robot Operating System*. " O'Reilly Media, Inc.", 2015.

[108] Gerardo Pardo-Castellote. Omg data-distribution service: Architectural overview. In *23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings.*, pages 200–206. IEEE, 2003.

[109] Joseph M Schlesselman, Gerardo Pardo-Castellote, and Bert Farabaugh. Omg data-distribution service (dds): architectural update. In *IEEE MILCOM 2004. Military Communications Conference, 2004.*, volume 2, pages 961–967. IEEE, 2004.

[110] Steve B Furber, Francesco Galluppi, Steve Temple, and Luis A Plana. The spinnaker project. *Proceedings of the IEEE*, 102(5):652–665, 2014.

[111] Manuel Huber, Daniel Pustka, Peter Keitler, Florian Echtler, and Gudrun Klinker. A system architecture for ubiquitous tracking environments. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 211–214. IEEE, 2007.

[112] Peter Keitler, Daniel Pustka, Manuel Huber, Florian Echtler, and Gudrun Klinker. Management of tracking for mixed and augmented reality systems. In *The Engineering of Mixed Reality Systems*, pages 251–273. Springer, 2010.

[113] Srdan Popić, Dražen Pezer, Bojan Mrazovac, and Nikola Teslić. Performance evaluation of using protocol buffers in the internet of things communication. In *2016 International Conference on Smart Systems and Technologies (SST)*, pages 261–265. IEEE, 2016.

[114] Huiyang Cui, Bin Zhang, Guohui Li, Xuesong Gao, and Yan Li. Contrast analysis of netconf modeling languages: Xml schema, relax ng and yang. In *2009 International Conference on Communication Software and Networks*, pages 322–326. IEEE, 2009.

[115] Martin Bjorklund. The yang 1.1 data modeling language. Technical report, 2016.

[116] Deepti Raghavan, Philip Levis, Matei Zaharia, and Irene Zhang. Breakfast of champions: towards zero-copy serialization with nic scatter-gather. In *Proceedings of the Workshop on Hot Topics in Operating Systems*, pages 199–205, 2021.

[117] Michael Wooldridge and Nicholas R Jennings. Intelligent agents: Theory and practice. *The knowledge engineering review*, 10(2):115–152, 1995.

[118] Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach. 2002.

[119] Nahal Norouzi, Gerd Bruder, Brandon Belna, Stefanie Mutter, Damla Turgut, and Greg Welch. A systematic review of the convergence of augmented reality, intelligent virtual agents, and the internet of things. *Artificial intelligence in IoT*, pages 1–24, 2019.

[120] Hyunji Chung and Sangjin Lee. Intelligent virtual assistant knows your life. *arXiv preprint arXiv:1803.00466*, 2018.

[121] Fei Tao and Meng Zhang. Digital twin shop-floor: a new shop-floor paradigm towards smart manufacturing. *Ieee Access*, 5:20418–20427, 2017.

[122] Fei Tao, He Zhang, Ang Liu, and Andrew YC Nee. Digital twin in industry: State-of-the-art. *IEEE Transactions on industrial informatics*, 15(4):2405–2415, 2018.

[123] Mengnan Liu, Shuiliang Fang, Huiyue Dong, and Cunzhi Xu. Review of digital twin about concepts, technologies, and industrial applications. *Journal of Manufacturing Systems*, 58:346–361, 2021.

[124] David Jones, Chris Snider, Aydin Nassehi, Jason Yon, and Ben Hicks. Characterising the digital twin: A systematic literature review. *CIRP Journal of Manufacturing Science and Technology*, 29:36–52, 2020.

[125] Michael Grieves and John Vickers. Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems. In *Transdisciplinary perspectives on complex systems*, pages 85–113. Springer, 2017.

[126] Concetta Semeraro, Mario Lezoche, Hervé Panetto, and Michele Dassisti. Digital twin paradigm: A systematic literature review. *Computers in Industry*, 130:103469, 2021.

[127] John Seely Brown, Allan Collins, and Paul Duguid. Situated cognition and the culture of learning. *Educational researcher*, 18(1):32–42, 1989.

[128] Keiichi Sato. Context sensitive interactive systems design: A framework for representation of contexts. In *Human-Centered Computing*, pages 1323–1327. CRC Press, 2019.

[129] Sean L Bowman, Nikolay Atanasov, Kostas Daniilidis, and George J Pappas. Probabilistic data association for semantic slam. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 1722–1729. IEEE, 2017.

[130] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017.

[131] Marius Fehr, Fadri Furrer, Ivan Dryanovski, Jürgen Sturm, Igor Gilitschenski, Roland Siegwart, and Cesar Cadena. Tsdf-based change detection for consistent long-term dense reconstruction and dynamic object discovery. In *2017 IEEE International Conference on Robotics and automation (ICRA)*, pages 5237–5244. IEEE, 2017.

[132] Tomu Tahara, Takashi Seno, Gaku Narita, and Tomoya Ishikawa. Retargetable ar: Context-aware augmented reality in indoor scenes based on 3d scene graph. In *2020 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pages 249–255. IEEE, 2020.

[133] Hugh Durrant-Whyte, David Rye, and Eduardo Nebot. Localization of autonomous guided vehicles. *Robotics Research*, pages 613–625, 1996.

[134] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052–1067, 2007.

[135] MWM Gamini Dissanayake, Paul Newman, Steve Clark, Hugh F Durrant-Whyte, and Michael Csorba. A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on robotics and automation*, 17(3):229–241, 2001.

[136] Philipp Fleck, Aimee Sousa Calepso, Sebastian Hubenschmid, Michael Sedlmair, and Dieter Schmalstieg. Ragrug: A toolkit for situated analytics. *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[137] Thomas Holz, Mauro Dragone, and Gregory MP O'Hare. Where robots and virtual agents meet. *International Journal of Social Robotics*, 1(1):83–93, 2009.

[138] Ghazanfar Ali, Hong-Quan Le, Junho Kim, Seung-Won Hwang, and Jae-In Hwang. Design of seamless multi-modal interaction framework for intelligent virtual agents in wearable mixed reality environment. In *Proceedings of the 32nd International Conference on Computer Animation and Social Agents*, pages 47–52, 2019.

[139] Thomas Plötz, Nils Y Hammerla, and Patrick L Olivier. Feature learning for activity recognition in ubiquitous computing. In *Twenty-second international joint conference on artificial intelligence*, 2011.

[140] Ruizhi Cheng, Nan Wu, Songqing Chen, and Bo Han. Reality check of metaverse: A first look at commercial social virtual reality platforms. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 141–148, 2022.

[141] Grant Kelly, Bruce McKenzie, et al. Security, privacy, and confidentiality issues on the internet. *Journal of Medical Internet Research*, 4(2):e861, 2002.

[142] Jaybie A De Guzman, Kanchana Thilakarathna, and Aruna Seneviratne. Security and privacy approaches in mixed reality: A literature survey. *ACM Computing Surveys (CSUR)*, 52(6):1–37, 2019.

[143] Peter Gabriel, M Bovenschulte, E Hartmann, W Groß, H Strese, KM Bayarou, M Haisch, M Mattheß, C Brune, H Strauss, et al. Pervasive computing: trends and impacts. *SecuMedia, Ingelheim*, 2006.

[144] Victoria Bellotti and Abigail Sellen. Design for privacy in ubiquitous computing environments. In *Proceedings of the Third European Conference on Computer-Supported Cooperative Work 13–17 September 1993, Milan, Italy ECSCW'93*, pages 77–92. Springer, 1993.

[145] Simson Garfinkel and Gene Spafford. *Web security, privacy & commerce.* " O'Reilly Media, Inc.", 2002.

[146] Dick Hardt. The oauth 2.0 authorization framework. Technical report, 2012.

[147] Natsuhiko Sakimura, John Bradley, Mike Jones, Breno De Medeiros, and Chuck Mortimore. Openid connect core 1.0. *The OpenID Foundation*, page S3, 2014.

[148] John Hughes and Eve Maler. Security assertion markup language (saml) v2. 0 technical overview. *OASIS SSTC Working Draft sstc-saml-tech-overview-2.0-draft-08*, 13, 2005.

[149] Michael Jones, John Bradley, and Nat Sakimura. Json web token (jwt). Technical report, 2015.

[150] Stian Thorgersen and Pedro Silva. *Keycloak-Identity and Access Management for Modern Applications.* Packt Publishing, 2021.

[151] Marcus A Christie, Anuj Bhandar, Supun Nakandala, Suresh Marru, Eroma Abeysinghe, Sudhakar Pamidighantam, and Marlon E Pierce. Using keycloak for gateway authentication and authorization. 2017.

[152] Sabrina Sicari, Alessandra Rizzardi, Luigi Alfredo Grieco, and Alberto Coen-Porisini. Security, privacy and trust in internet of things: The road ahead. *Computer networks*, 76:146–164, 2015.

[153] Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. Robust physical-world attacks on machine learning models. *arXiv preprint arXiv:1707.08945*, 2(3):4, 2017.

[154] Vincent C Hu, David Ferraiolo, Rick Kuhn, Arthur R Friedman, Alan J Lang, Margaret M Cogdell, Adam Schnitzer, Kenneth Sandlin, Robert Miller, Karen Scarfone, et al. Guide to attribute based access control (abac) definition and considerations (draft). *NIST special publication*, 800(162):1–54, 2013.

[155] Vincent C Hu, D Richard Kuhn, David F Ferraiolo, and Jeffrey Voas. Attribute-based access control. *Computer*, 48(2):85–88, 2015.

[156] Daniel Servos and Sylvia L Osborn. Current research and open problems in attribute-based access control. *ACM Computing Surveys (CSUR)*, 49(4):1–45, 2017.

[157] Roger Clarke. Information technology and dataveillance. *Communications of the ACM*, 31(5):498–512, 1988.

[158] Lucio La Cava, Sergio Greco, and Andrea Tagarelli. Understanding the growth of the fediverse through the lens of mastodon. *Applied Network Science*, 6(1):1–35, 2021.

[159] Óscar Blanco-Novoa, Paula Fraga-Lamas, Miguel A Vilar-Montesinos, and Tiago M Fernández-Caramés. Creating the internet of augmented things: An open-source framework to make iot devices and augmented and mixed reality systems talk to each other. *Sensors*, 20(11):3328, 2020.

[160] Kristoffer Waldow and Arnulph Fuhrmann. Using MQTT for platform independent remote mixed reality collaboration. *Mensch und Computer 2019-Workshopband*, 2019.

[161] Elad Michael, Tyler Summers, Tony A Wood, Chris Manzie, and Iman Shames. Probabilistic data association for semantic slam at scale. *arXiv preprint arXiv:2202.12802*, 2022.

[162] J Edward Swan. The replication crisis in empirical science: Implications for human subject research in mixed reality. In *2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pages xxxvi–xxxvi. IEEE, 2018.

[163] Nikkan Kogyo Shimbun. *Poka-yoke: improving product quality by preventing defects*. Crc Press, 1989.

[164] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[165] Christian Eichhorn, Adnane Jadid, David A Plecher, Sandro Weber, Gudrun Klinker, and Yuta Itoh. Catching the drone-a tangible augmented reality game in superhuman sports. In *2020 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pages 24–29. IEEE, 2020.

[166] Christian Eichhorn, Martin Lurz, David A Plecher, Sandro Weber, Monika Wintergerst, Birgit Kaiser, Sophie L Holzmann, Christina Holzapfel, Hans Hauner, Kurt Gedrich, et al. Inspiring healthy food choices in a virtual reality supermarket by adding a tangible dimension in the form of an augmented virtuality smartphone. In *2021 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 548–549. IEEE, 2021.

[167] David A Plecher, Christian Eichhorn, Annette Köhler, and Gudrun Klinker. Oppidum-a serious-ar-game about celtic life and history. In *Games and Learning Alliance: 8th International Conference, GALA 2019, Athens, Greece, November 27–29, 2019, Proceedings 8*, pages 550–559. Springer, 2019.

[168] David A Plecher, Annalena Ulschmid, Tim Kaiser, and Gudrun Klinker. Projective augmented reality in a museum: development and evaluation of an interactive application. 2020.

[169] David Plecher, Maximilian Ludl, and Gudrun Klinker. Designing an ar-escape-room with competitive and cooperative mode. In *GI VR/AR Workshop*. Gesellschaft für Informatik eV, 2020.

[170] Valeriya Lehrbaum, Asa MacWilliams, Joseph Newman, Nischita Sudharsan, Seongjin Bien, Konstantin Karas, Chloe Eghtebas, Sandro Weber, and Gudrun Klinker. Enabling customizable workflows for industrial ar applications. 2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR) [unpublished at time of writing], 2022.

[171] David L Mills. Internet time synchronization: the network time protocol. *IEEE Transactions on communications*, 39(10):1482–1493, 1991.

[172] Steve T Watt, Shankar Achanta, Hamza Abubakari, Eric Sagen, Zafer Korkmaz, and Husam Ahmed. Understanding and applying precision time protocol. In *2015 Saudi Arabia Smart Grid (SASG)*, pages 1–7. IEEE, 2015.

[173] Michalis Vrigkas, Christophoros Nikou, and Ioannis A Kakadiaris. A review of human activity recognition methods. *Frontiers in Robotics and AI*, 2:28, 2015.