Penser le réseau comme un bien commun

Jean-Marie Dallet

L'art en réseau, apparu dans la seconde moitié des années 1990, est un art qui utilise le réseau internet comme médium. Les œuvres interactives qui s'en revendiquent sont des créations conçues « **par** » et « **pour** » le réseau internet. « **Par** », parce que formellement le réseau influence la conception même de l'œuvre (langage web, hyperliens, protocoles, participation en ligne, etc.). « **Pour** », parce que du point de vue du sens, elles n'existent que dans cet espace.

Si une partie de l'art en réseau relève de la création d'objets interactifs « finis » qui usent des qualités de **dissémination** et de **dématérialisation** propre à l'internet — cet art a déjà connu de nombreuses dénominations : « Art Internet », « Art réseau », « Cyberart » ou encore « Web art », avant que l'appellation « Net art » s'impose —, une autre partie de l'art en réseau s'intéresse à la création de performance en temps réel ou les questions de **temporalité** et de **réflexions sur les médias** s'avèrent fondamentales.

C'est de cette seconde catégorie de réalisation que cet article va parler. Pour faire un peu d'archéologie des médias, c'est au début des années 1980, avec *Hole in Space* de **Kit Galloway** et **Sherrie Rabinowitz**, dispositif qui a permis pendant deux heures, trois soirs durant, aux habitants de New York de rencontrer en direct ceux de Los Angeles via des transmissions satellites, qu'un art en ligne est né du désir, par les artistes, de s'approprier les moyens des médias de communication.

L'artiste anglais **Paul Sermon** poursuivra, la décennie suivante, cet art de la téléprésence. Dans les dispositifs *Telematic Dreaming* (1992) et *Telematic Vision* (1992), par exemple, qui ont durablement orienté l'esthétique de ses dispositifs ultérieurs, un système de **visioconférence sur fibre optique** reliait deux lits ou deux canapés installés dans des galeries éloignées. La liaison visuelle en temps réel permettait aux visiteurs de se voir à distance et, surtout, d'éprouver l'illusion de partager un même lit ou un même canapé, projetés dans le « troisième espace » de l'image vidéo.

Paul Sermon, Telematic Dreaming, 1994. Image projetée de Susan Kozel qui interagit avec un participant de la galerie, exposition Ik + De Ander, Amsterdam, 1994.

Ce désir de rencontres et d'échanges, de jeu aussi où se mêlent maîtrise des outils, des temps, des synchronicités à l'aléatoire le plus complet concernant « ce qui se passe ailleurs et que je ne peux contrôler directement » est au centre de l'intérêt que les artistes portent à ces technologies qui usent du réseau comme d'autres du pinceau.

Ne soyons pas naïf cependant, car ces technologies de téléprésence sont complexes et demandent pour être déployées des connaissances tant sur l'architecture des réseaux, que sur les langages de programmation, et les techniques qu'elles soient analogiques ou numériques.

Je m'appuierai sur trois exemples pour toucher du doigt cette intrication des complexités et essayer d'en rendre lisibles les strates :

1. La performance musicale *Hermès v2* dans laquelle son créateur João Svidzinski invente une logique et donc un programme pour mettre en scène les spectateurs dans une performance musicale multijoueurs ;

2. La création d'outils artistiques pour la performance réseau par le collectif Apo33 qui nécessite la création d'environnements de travail numérique entièrement maitrisables, c'est-à-dire indépendants des systèmes commerciaux ;

3. La plateforme mmmap (marcel multimedia art project), programme politique dont l'ambition est de proposer un écosystème de travail et de création autonome pour des artistes et des chercheurs à l'échelle mondiale.

1. Performance musicale *Hermès v2*

João Svidzinski est compositeur, réalisateur en informatique musicale et enseignant-chercheur en université. Il développe depuis 2021, un système de performance musicale en réseau local, ***Hermès v2***, qu'il avait présenté en 2022 au colloque international "Arts en réseau", événement soutenu par la Région et la DRAC Nouvelle-Aquitaine, l'Institut ACTE, Paris 1 Panthéon-Sorbonne et La Métive (Centre international de résidences d'artistes, Moutier-d'Ahun).

À cette occasion, les participants étaient répartis en deux groupes : les « auditeurs », qui se limitaient à l'écoute, et les « actifs », qui, via smartphone, tablette ou ordinateur portable, se connectaient à une interface graphique dans un navigateur web et contribuaient ainsi directement à l'exécution de l'œuvre.

L'environnement de jeu s'ouvrait sur un écran d'accueil **dynamique et interactif**, qui évoluait au fil de la pièce. Des éléments de l'interface graphique (GUI), tels que des boutons ou des zones de texte, apparaissaient puis s'effaçaient durant la performance. Le dispositif assumait ainsi un double rôle : **transmettre les consignes** aux participants et **orchestrer leurs interactions** avec l'électronique.

Pendant la performance, le compositeur pilotait un ordinateur hôte relié au réseau Wi-Fi. Cette machine rassemblait tout le traitement numérique d'Hermès v2 : les serveurs de communication, les gestionnaires de données et le moteur DSP. Les serveurs assuraient les échanges réseau, les gestionnaires de données traitaient les paramètres de contrôle, et le DSP prenait en charge la génération, la transformation et la spatialisation du son.

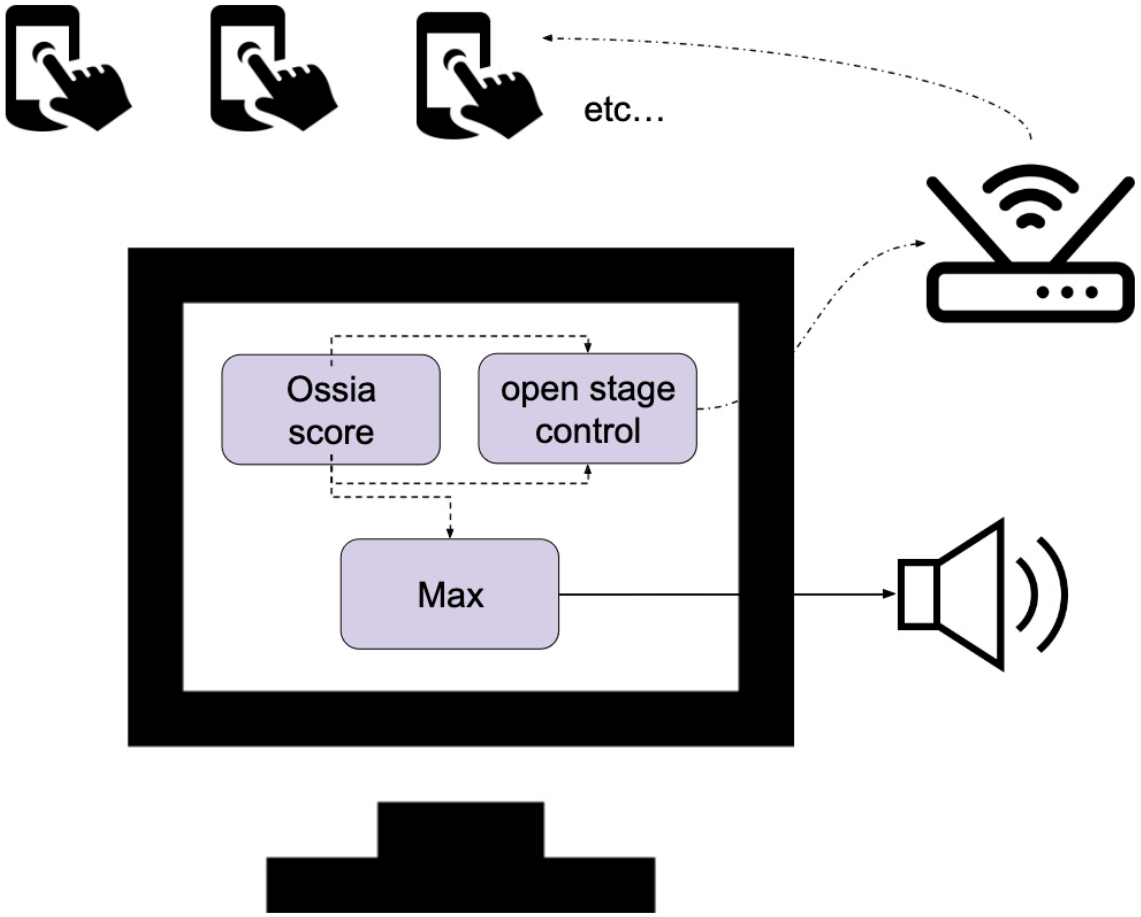


Schéma technique de la version réseau local d'*Hermès v2.* L'ordinateur hôte est représenté au centre. Trois logiciels y tournent : Ossia score, Open Stage Control et Max. Tous trois communiquent entre eux. Max, responsable de la génération et manipulation du son, envoie les signaux audio vers les haut-parleurs. Open Stage Control reçoit et envoie, via le point d'accès wifi, des paramètres de contrôle de chaque participant et gère également Ossia Score. Image © J. Svidzinski

Trois logiciels structuraient le dispositif : 1. **Open Stage Control** pour le pilotage de l'interface graphique et la mise en réseau des terminaux mobiles avec l'hôte ; 2. **Ossia Score** pour la conduite temporelle et l'orchestration des liaisons entre logiciels ; 3. **Max** pour le traitement audionumérique.

Détaillons une séquence : tout d'abord **Ossia Score** ordonne l'affichage des GUI — quatre boutons, un texte et des *faders* — qu'**Open Stage Control** rend ensuite à l'écran. Quand un participant appuie sur un bouton, son appareil, via le client **Open Stage Control**, envoie un message à **Ossia Score**, qui le relaie vers **Max** et déclenche un échantillon sonore.

Le traitement sonore issu de ces interactions s'appuyait sur **Max**, enrichit d'objets **Faust** — en particulier le ***Granulateur*** et ***l'Harmonizer*** — exploitant le principe de la réinjection matricielle.

Faust, ou *Functional Audio Stream*, est un langage de programmation fonctionnel conçu pour la synthèse sonore et le traitement audio. Développé par le département de recherche du GRAME-CNCM, il est utilisé pour des applications de traitement du signal à haute performance et la création de *plug-ins* audio compatibles avec diverses plateformes et standards.

Le contrôle était assuré par **Antescofo**, qui permet à la fois la manipulation et la transmission de paramètres. Doté d'une syntaxe spécifique, proche des langages de programmation classiques, il combine plusieurs paradigmes pour un pilotage souple et précis des processus sonores.

Enfin, le rendu multicanal a été réalisé avec la bibliothèque **HOA** (High Order Ambisonics) et diffusé dans la salle. Cette bibliothèque propose un large éventail de traitements spatiaux audionumériques, permettant de positionner et de transformer le son dans l'espace.

2. Création d'outils artistiques pour la performance réseau

D'autres artistes à l'instar du **collectif Apo33** basé à Nantes développent, depuis le début des années 2000, des outils complexes à même de configurer toute une chaîne d'outils de créations audiovisuelles.

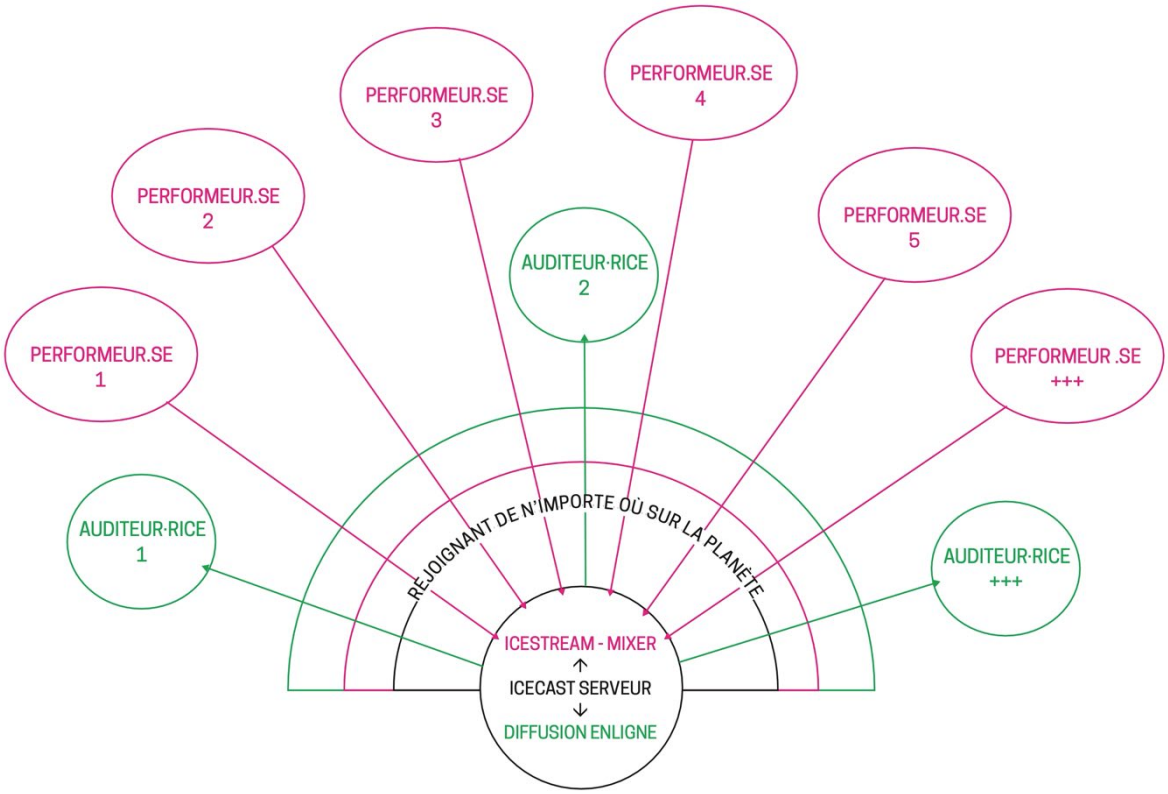
Tout d'abord, pour ne pas dépendre des évolutions des systèmes d'exploitation (OS) — ensemble de programmes qui gèrent les ressources d'un ordinateur — commercialisés par Windows ou Apple, par exemple, Apo33 a conçu son propre OS (distribution GNU/Linux), **Apodio** (https://sourceforge.net/projects/apodio/), spécialisé dans les tâches de productions audiovisuelles en direct.

La gestion de ces flux vidéo et audio était réalisée dans un premier temps par le programme **Icestream** qui jouait le rôle d'une brique de diffusion et constituait, utilisé avec **Apodio**, un environnement unifié, reproductible et indépendant des plateformes propriétaires.

Icestream (https://apo33.org/?p=587) a évolué depuis et se présente maintenant comme un logiciel serveur de *streaming* multimédia open-source qui permet de mixer, recevoir ou envoyer des flux audio et vidéo à travers un réseau local ou Internet, utilisant les technologies de flux en direct : format de transmission de données **OSC** (Open Sound Control) via le **protocole FTP** (File Transfer Protocol), le ***streaming live*** à partir de 2005.

Disposer de serveurs autonomes autorise le développement d'outils dédiés aux œuvres, tout en garantissant leur maintenance et leur évolutivité dans la durée. Observons le logiciel **GIASO** (Great International Audio Streaming Orchestra) qui constitue une plateforme orchestrale internationale en ligne dédiée à la performance en réseau.

GIASO (https://apo33.org/?p=253) s'appuie sur une **plateforme multiplexe bidirectionnelle** permettant la performance et le mixage **en temps réel** de flux audio hétérogènes. Les *streams*, émis depuis des sites multiples, sont **agrégés** puis **recomposés** dans un espace de diffusion unique. La **spatialisation** est assurée par un système de transmission **IP libre** et **multiflux** qui orchestre le routage et la distribution des signaux dans l'espace virtuel de la performance, produisant une immersion de grande complexité.



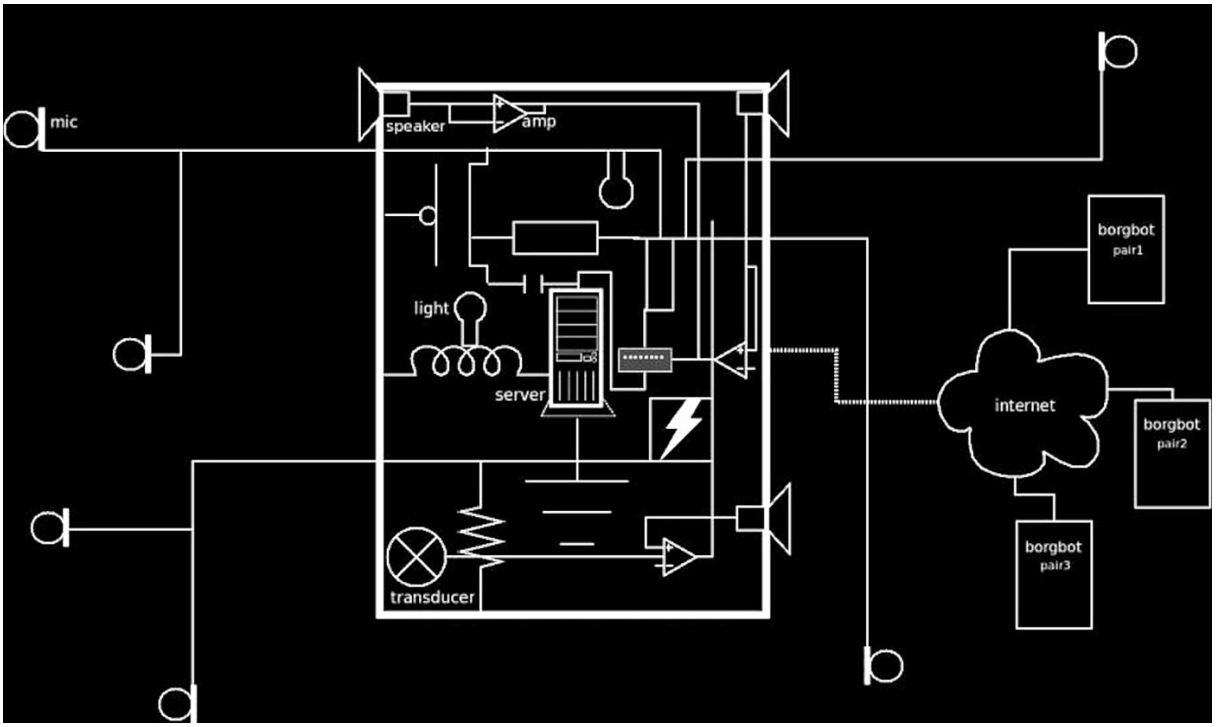
Apo33, *GIASO (Great International Audio streaming Orchestra)*, schéma. Image © collectif Apo33

Comme le rapportent Julien Ottavi et Jenny Pickett d'Apo33 : « Dans cet orchestre numérique, les artistes sont transformés en entités dématérialisées et chacun agit comme un nœud dans un réseau plus vaste de performeurs et d'explorateurs audio. Ce cadre unique permet une nouvelle forme de composition orchestrale dans laquelle la création musicale émerge dynamiquement d'une communauté interconnectée plutôt que d'une pratique située et/ou isolée. »

Ce type d'outils de partage et d'actions sur des flux audio, ils l'étendront à des applications capables de gérer le son bien sûr, mais encore des données renvoyées par des capteurs. Ainsi, avec l'outil **Bots** (2009), le territoire devient l'espace sensible sur lequel un ensemble de modules autonomes, les « bots », implantés *in situ* (ville/nature), sont chargés d'explorer les propriétés actives des lieux et de les interconnecter.

Chaque module capte des données de son environnement (son, mouvement, lumière, etc.) et les envoie sur Internet sous forme de flux : 1. **flux audio** : le son capté sur place ; 2. **flux de données** : mesures de capteurs qui servent à piloter le son.

Ces données contrôlent des traitements réalisés dans des patches **Pure Data**. Les patches, reliés entre eux, mélangent les différents sons, les transforment et recomposent en continu le paysage sonore. Le résultat est un univers audio évolutif, produit par une chaîne où les données de chaque site modifient les traitements, qui à leur tour transforment les sons transmis.



Apo33, *The Borgbot: the Dawn of Machinity*, version Alpha, schéma. Projet collaboratif initié par Apo33. Environnement sonore intégré. Image © collectif Apo33

Le dispositif fonctionne de manière continue : on peut écouter le ***live stream*** au quotidien et, si on le souhaite, ajuster des paramètres. La musique garde ainsi une certaine autonomie, tout en laissant une place à l'écoute et à l'intervention ordinaire de l'utilisateur.

3. Plateforme mmmap (marcel multimedia art project) :

Considérer le réseau comme un espace commun à réinventer nous a amenés avec le collectif **MARCEL** (https://www.mmmarcel.org/start), sous la houlette du théoricien des médias **Don Foresta**, à réfléchir à un dispositif technique qui permettrait à des artistes de collaborer lors de performances à distance.

1.1 Architecture serveur :

Ce dispositif technique prend s'appuie sur une **architecture mesh** pour les serveurs (*mesh network architecture*) qui présente une organisation horizontale dans laquelle chaque serveur (ou nœud) est connecté à d'autres serveurs, sans dépendre donc d'un point central. Ce type d'organisation, plutôt qu'en étoile ou circulaire, présente l'avantage d'offrir une très grande résilience — en cas de coupure du lien entre deux serveurs, par exemple — et un routage flexible — les serveurs communiquant régulièrement entre eux adaptent automatiquement le trajet de l'information pour optimiser la circulation des flux de données.

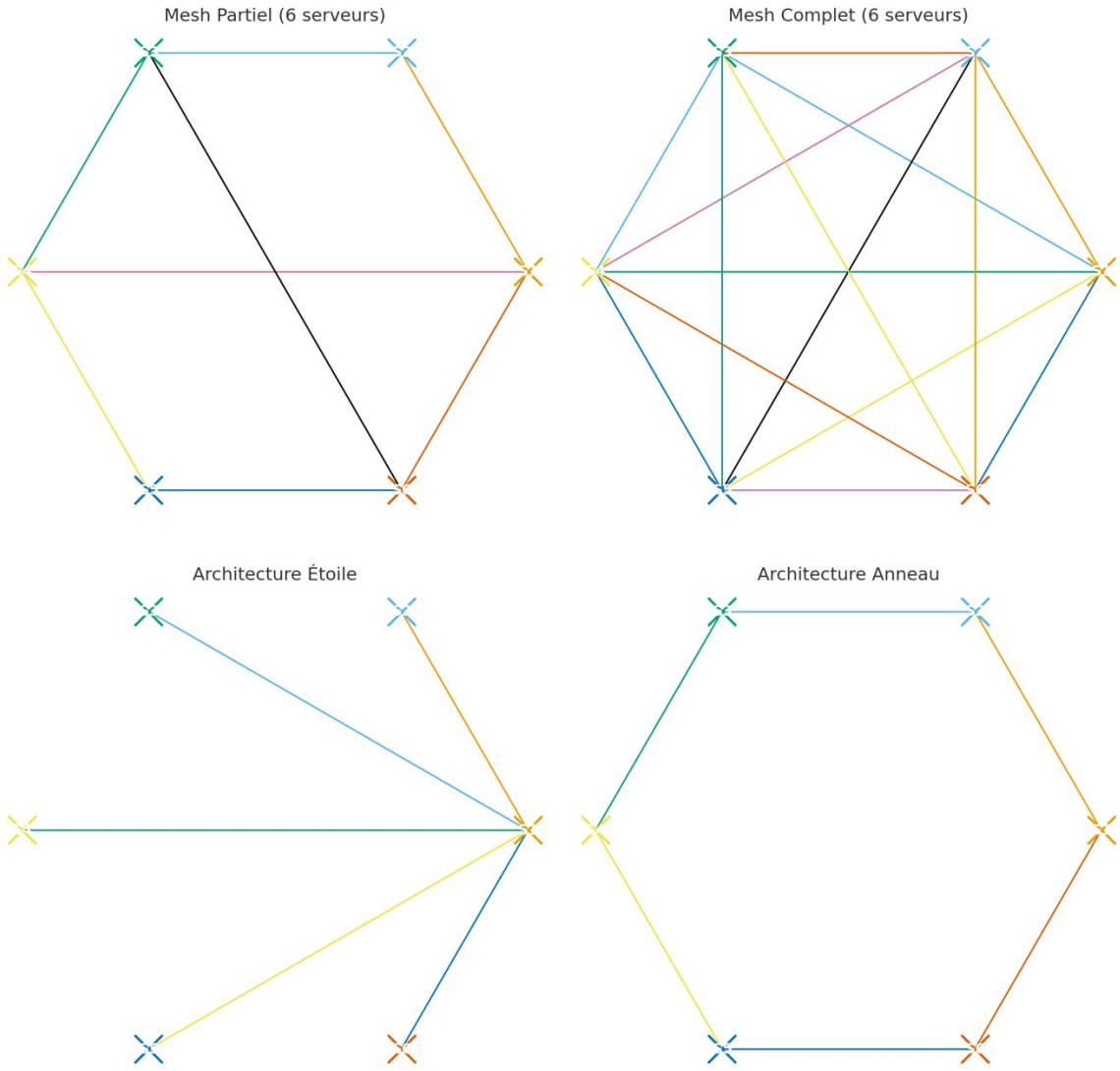


Image : source ChatGPT

Le **logiciel serveur de mmmap** (marcel multimedia art project) est développé par l'artiste et intermittent du spectacle **Benoît Lahoz**, interrogé pour qu'il détaille la logique de fonctionnement de mmmap et des outils qu'il a utilisés, adaptés et développés.

1.2 Architecture logicielle :

L'architecture logicielle retenue est celle dite de **microservices** qui découpe les services permis par l'application en de multiples petits services et les distribue sur les serveurs les plus puissants du réseau mis en place lors des performances. L'avantage d'un tel système par rapport à une architecture en monolithe — tous les services regroupés sur un même serveur ? Une agilité, une résilience, une scalabilité fine et la possibilité d'effectuer des modifications sur chaque service sans être obligé de réécrire toute l'application. L'inconvénient est qu'il faut un outil d'orchestration permettant une bonne gestion des flux et que le dispositif, en multipliant les échanges entre les machines, augmente mécaniquement les effets de latence. On perd du temps et donc de la synchronicité entre les événements, ce qui peut se révéler problématique lors des performances musicales ou théâtrales. Cependant, ce dernier inconvénient est en partie contrebalancé par le partage de la bande passante entre les serveurs qui fluidifie le transport des informations : images, voix, musiques, textes, etc.

Il faudrait ajouter un autre avantage d'importance à ceux cités plus haut, c'est la possibilité de pouvoir travailler avec des serveurs de taille et de coût très différents, du RaspberryPi, par exemple, au cluster de serveurs universitaires ou industriels. Cela facilite l'installation du dispositif qui ne nécessite pas de s'appuyer sur des infrastructures lourdes.

1.3 Utilisateur :

Du côté de l'utilisateur, l'interface graphique est développée par plugin grâce au framework Vue.js. L'interface permet de créer un espace, une salle commune à un projet, qui met à disposition des autres usagers les signaux générés localement :

1. Vidéo : vidéo encodée ; issue du partage d'écran ; tableau blanc ; tout type d'image générée soit par des outils offerts par la salle elle-même (Pure Data, Processing.js, par exemple), soit par des outils locaux (Max MSP/Jitter, Vuo, VDMX, Resolume, etc.) via une application native.
2. Son : signal audio synchronisé, direct depuis un ou plusieurs micros (monocanal ou multicanaux) ; signal audio généré sur le serveur ou envoyé depuis une application locale (Ableton Live, etc.) via une application native.
3. Données : tous types de données. Cela peut aller du partage de fichiers aux données renvoyées par des capteurs, qui peuvent éventuellement ensuite être transformées en données utilisables localement : DMX pour la lumière, OSC3 pour le contrôle, etc.).

Comme le souligne Lahoz : « Les signaux envoyés au serveur par chacun des utilisateurs de la salle commune en question sont, en quelque sorte, mis dans un pot commun. Chacun est alors libre de récupérer le signal brut d'un autre utilisateur pour son propre usage local. Et surtout, il est possible pour les utilisateurs de la salle de travailler sur un même patch (pour reprendre la terminologie de Pure Data) avec tout ou partie des signaux disponibles (entrées). »

1.4 Langages :

Les langages utilisés pour le développement sont les suivants :

1.4.1 côté serveur :

– Javascript côté serveur (Node.js) ;

– C / C++ / Python *bindings* : il s'agit de l'interface entre des librairies programmées en différents langages et le Javascript.

1.4.2 côté client :

– Javascript via le framework Vue.js ;

– Pug, Stylus pour l'écriture des pages elles-mêmes.