IL PROTOCOLLO OSC

OpenSoundControl (OSC) è una specifica di trasporto dati (una codifica) per la comunicazione in tempo reale tra applicazioni e hardware attraverso una rete TCP/IP o internet. È stato sviluppato dai ricercatori Matt Wright e Adrian Freed al Center for New Music & Audio Technologies (CNMAT).

..è un protocollo *aperto*: lo spazio degli indirizzi è interamente definito dall'utente, consentendogli di essere sia leggero che infinitamente personalizzabile ed estensibile alle esigenze specifiche dell'utente. I messaggi OSC sono differenziati tra loro da uno schema di denominazione simbolica in stile URI che consente un'organizzazione gerarchica dello spazio degli indirizzi

..è un protocollo *accurato*: i messaggi supportano una vasta gamma di tipi di dati simbolici e ad alta risoluzione. Possono incorporare marcatori temporali ad alta risoluzione per una coordinazione temporale precisa e possono essere assemblati in pacchetti per garantire consegna ed esecuzione simultanee

. . . è un protocollo *flessibile*: include un linguaggio di corrispondenza di modelli per specificare più destinatari di un singolo messaggio. È progettato per massima interoperabilità e quindi fornisce una soluzione pronta per comunicare tra applicazioni e dispositivi, sia localmente che su una rete

Trasmissione

Il protocollo OSC supporta un'architettura **client/server**, in cui un client invia dati (organizzati in pacchetti) messaggi a un server. Una trasmissione OSC (nella maggior parte dei casi) richiede un indirizzo IP e un numero di host per la connessione: client e server devono devono avere una porta di uscita e entrata; i client, devono indicare gli indirizzi IP dei server in modo da poter identificare la destinazione per i loro pacchetti.
È bene sottolineare che Open Sound Control è un protocollo di alto livello, ovvero, non si occupa del meccanismo di trasmissione dei dati. La rete che trasporta un pacchetto OSC è responsabile della consegna sia del contenuto che del dato riguardante la sua grandezza.

La struttura del formato di uno stream OSC è organizzata in *bundle*, che a loro volta possono contenere messaggi o altri bundle. Ogni bundle è costituito da una sequenza di elementi definiti con un time tag, che rappresenta un momento nel tempo in cui i messaggi all'interno del bundle devono avere effetto.

1. **Bundle OSC**: È l'elemento fondamentale dello stream OSC. Ogni bundle ha una struttura del genere:
    - **Intestazione**: Composta dall'identificatore di stringa OSC "#bundle" e un time tag, che indica il momento in cui il bundle dovrebbe essere processato.
    - **Elementi del Bundle**: Possono essere messaggi OSC o ulteriori bundle. Questi elementi vengono elencati uno dopo l'altro all'interno

del bundle.

È una sequenza di messaggi e/o bundle. Tutti i messaggi nello stesso bundle devono essere elaborati dal server OSC atomicamente, in altre parole è come se tutti i messaggi nel bundle fossero trattati in un solo istante, come una sola entità. Un pacchetto OSC può essere un bundle o un unico messaggio. Ogni bundle ha un "time tag" che si occupa della sincronizzazione dei singoli messaggi contenuti in esso e il suo formato è quello utilizzato dal Network Time Protocol Internet (NTP).

2. **Messaggio OSC**: È l'unità di base dei dati OSC e contiene le informazioni per eseguire un'azione specifica. Ogni messaggio è composto da:
   - **Indirizzo del pattern OSC**: È una stringa che inizia con una barra (/) e rappresenta il percorso dell'indirizzo nel modello gerarchico degli indirizzi OSC.
   - **Stringa type tag OSC**: Una stringa che inizia con una virgola (,) e definisce i tipi di dati dei successivi argomenti OSC nel messaggio. I diversi tipi di dati supportati dagli Argomenti OSC sono int32, float32, stringhe ASCII e blob. Altri tipi di dati supportati da alcune implementazioni OSC sono numeri a 64 bit, colore RGBA, 'True' e 'False'. Segiue la tabella di corrispondenza type tag - argomento:

Type tag	Argument
i	int32
f	float32
s	OSC string
b	OSC blob

   - **Argomenti OSC**: Sono i dati veri e propri che vengono trasmessi nel messaggio, e il loro tipo è definito dalla stringa di tag di tipo OSC.
3. **Time Tag OSC**: È un elemento opzionale presente all'interno di un bundle che specifica un momento temporale in cui i messaggi all'interno del bundle devono essere processati. Un server OSC deve avere accesso a una corretta rappresentazione del tempo attuale in assoluto. OSC non fornisce alcun meccanismo per la sincronizzazione del clock. Quando un pacchetto OSC contenente un singolo messaggio viene ricevuto, il server OSC dovrebbe invocare il corrispondente OSC Method immediatamente, vale a dire, il più presto possibile dopo la ricezione del pacchetto. Diversamente, quando viene ricevuto un pacchetto OSC contenete un OSC Bundle è il Time Tag OSC di quest'ultimo che determina quando gli OSC Method corrispondenti agli OSC Messages (contenuti nell'OSC Bundle) debbano essere invocati. Se il tempo rappresentato dal Time Tag OSC è precedente o uguale al tempo attuale, il server di OSC dovrebbe invocare gli OSC Method immediatamente (salvo che l'utente non abbia configurato il server OSC

affinché elimini i messaggi che arrivino troppo tardi). Nel caso contrario che il Time Tag OSC rappresenti un momento nel futuro, il server OSC deve memorizzare il pacchetto OSC fino al momento specificato e solo allora invocare gli OSC Methods appropriati. I Time Tag OSC sono rappresentati da un numero a 64 bit a virgola fissa (fixed-point). I primi 32 bits specificano il numero di secondi dalla mezzanotte del primo gennaio del 1900, mentre i successivi 32 bits specificano la parte frazionaria di secondo con una precisione di circa 200 picosecondi. Questa è la rappresentazione usata dal protocollo NTP in internet. Se il valore del Time Tag è costituito da 63 bits a zero e seguito da uno nella posizione del meno significativo, allora questo è il caso speciale che significa "immediatamente". Gli OSC Messages nello stesso OSC Bundle sono "atomici"; i loro OSC Methods corrispondenti devono essere invocati in immediata successione, nessun altro tipo di elaborazione deve aver luogo tra le varie invocazioni degli OSC Methods. Quando un OSC Address Pattern è inviato a multipli OSC Methods, l'ordine nel quale i corrispondenti OSC Methods sono invocati non è specificato. Quando un pacchetto OSC contiene diversi OSC Messages, il set di OSC Methods corrispondenti agli OSC Messages deve essere invocato rispettando lo stesso ordine di successione dei messaggi OSC all'interno del pacchetto. Quando i Bundles contengono altri Bundles, il Time Tag OSC del Bundle più interno (racchiuso) deve essere più grande o quanto meno uguale a quello del Bundle più esterno (che racchiude). Il requisito di "atomicità" per gli OSC Messages in uno stesso Bundle non è applicata a quelli del Bundle di livello inferiore (racchiuso).

4. **Queries**: Le queries sono messaggi OSC che chiedono al server di rimandare delle informazioni al client.

Questa struttura consente di organizzare e trasmettere dati di vario tipo in modo flessibile e efficiente all'interno di uno stream OSC, consentendo il controllo temporale e la trasmissione di informazioni tra dispositivi e applicazioni in tempo reale.

Ogni Server OSC ha un set di OSC Methods. Questi si riferiscono alle azioni o alle funzioni che vengono invocate o eseguite quando un messaggio OSC specifico viene ricevuto o elaborato da un'applicazione o da un dispositivo che supporta il protocollo OSC. Quando un messaggio OSC arriva a un destinatario, l'indirizzo del pattern OSC e i relativi argomenti vengono analizzati per determinare quale azione o funzione deve essere attivata. Questa funzione o azione associata a un particolare indirizzo OSC è comunemente chiamata *metodo OSC*.
Tali metodi sono organizzati in una struttura ad albero chiamata *OSC address space* in cui i nodi sono detti *OSC container* e le foglie i *methods*: L'OSC Address di un OSC Method è un nome simbolico che dà il percorso completo all'OSC Method nell'OSC Address Space, partendo dalla radice dell'albero. Un OSC Address di un OSC Method inizia con il carattere "/" (forward slash), seguito dai nomi di tutti gli OSC Containers, in ordine, lungo il percorso dalla radice dell'albero all'OSC Method separati da caratteri slash, infine troviamo il

nome dell'OSC Method. La sintassi degli indirizzi OSC è stata scelta in modo che corrisponda alla sintassi degli URL. es. *sintesi/additiva/sine*.
Regole:
Quando un Server OSC riceve un OSC Message, questi deve invocare l'appropriato OSC Method nel suo OSC Address Space basato sull'OSC Address Pattern dell'OSC Message. Questo processo è chiamato dispatching (spedizione) dell'OSC Message all'OSC Methods che farà il match (la corrispondenza) con il suo OSC Address Pattern. Tutti gli OSC Methods che trovano corrispondenza sono invocati con lo stesso argomento, cioè con gli OSC Arguments degli OSC Message. Le parti di un OSC Address o di un OSC Address Pattern sono le substringhe" tra coppie adiacenti di slash o la "substringa" dopo l'ultimo slash. Un OSC Message ricevuto deve essere spedito a ogni OSC Method nel corrente OSC Address Space il quale OSC Address corrisponderà all'OSC Address Pattern dell'OSC Message. Un OSC Address Pattern corrisponderà a un OSC Address se:

1. l'OSC Address e l'OSC Address Pattern contengono lo stesso numero di parti;
2. ogni parte dell'OSC Address Pattern corrisponde con la parte corrispondente dell'indirizzo OSC.

Una parte di un OSC Address Pattern corrisponde a una parte di un OSC Address se ogni carattere consecutivo nell'OSC Address Pattern corrisponde alla successiva consecutiva substringa dell'OSC Address e ogni carattere nell'OSC Address è corrispondente a qualcosa nel OSC Address Pattern. Queste sono le regole di corrispondenza per i caratteri negli OSC Address Pattern:

1. '?' negli OSC Address Pattern fa il matching di ogni singolo carattere;
2. '*' negli OSC Address Pattern fa il matching di ogni sequenza di zeri o più caratteri;
3. Una stringa di caratteri tra parentesi quadre (es. [string]") negli OSC Address Pattern fa il matching di ogni carattere nella stringa. All'interno delle parentesi quadre, il segno meno (-) e il punto esclamativo (!) hanno un significato particolare:
   - Due caratteri separati da un segno meno indicano un range di caratteri che corrisponde a quelli compresi tra i due in sequenza. (es. a-d equivale a "a,b,c,d")
   - Un punto esclamativo all'inizio di una stringa racchiusa tra parentesi nega il senso della lista, in altre parole significa che la lista fa il matching di ogni carattere che non è nella lista.
4. Una lista separata da virgole in una stringa all'interno di parentesi graffe (es."{foo,bar}") negli OSC Address Pattern fa il matching di ogni stringa nella lista.
5. Ogni altro carattere in un OSC Address Pattern fa il matching solo dello stesso carattere.

UDP/TCP

Il protocollo TCP (Trasmission Control Protocol) è uno dei principali componenti della grande famiglia dei protocolli internet. È un accordo standardizzato per la trasmissione di dati tra utenti di una rete informatica.
I sistemi che comunicano mediante TCP godono di un servizio *full-duplex* con conferma e controllo di flusso (le eventuali perdite di dati vengono riconosciute e corrette automaticamente; per questo, il TCP viene denominato anche protocollo affidabile): possono inviare e ricevere contemporaneamente dati.
Le unità di trasmissione fondamentali utilizzate dal protocollo sono i segmenti (pacchetti) che, oltre al messaggio effettivo, possono contenere anche informazioni di controllo e sono limitati a una dimensione di 1.500 byte.

La procedura concreta di instaurazione della connessione con il protocollo TCP è la seguente:

1. Nel primo passaggio, il client che richiede la connessione invia al server un pacchetto SYN o segmento SYN (dall'inglese synchronize = *sincronizzare*) con un numero sequenziale individuale e casuale. Questo numero assicura la trasmissione completa nella sequenza corretta (senza doppioni).
2. Dopo che il server ha ricevuto il segmento, acconsente all'instaurazione della connessione restituendo un pacchetto SYN-ACK (dall'inglese acknowledgement = *conferma*), comprensivo del numero sequenziale del client aumentato di 1. Inoltre, trasmette al client il proprio numero sequenziale.
3. Infine, il client conferma la ricezione del segmento SYN-ACK inviando un proprio pacchetto ACK che, in questo caso, contiene il numero sequenziale del server aumentato di 1. Al contempo può già trasferire i primi dati al server.

Entrambi i partecipanti alla comunicazione possono interrompere una connessione TCP instaurata; in più, la connessione può anche essere interrotta unilateralmente. Quest'ultimo caso viene denominato connessione chiusa per metà; in questa situazione, la controparte è ancora autorizzata a trasmettere dati anche se l'altro partecipante ha già interrotto la connessione.

Differentemente dal TCP, il protocollo UDP (User Datagram Protocol) fornisce un servizio connectionless: non garantisce né l'arrivo dei pacchetti a destinazione, né la ritrasmissione in caso di perdita d'informazione, né la corretta sequenza. I dati vengono trasmessi in pacchetti chiamati datagram di dimensioni prestabilite. Dall'altro verso, però, tali differenze rendono l'UDP molto più veloce ed efficiente, mancando delle operazioni di riordino e ritrasmissione: è un protocollo *stateless*, non tiene conto dello stato della connessione.

References

1. OpenSoundControl specification, https://opensoundcontrol.stanford.edu/index.html

2. OSC – Open Sound Control (parte 1) https://www.ageofaudio.com/osc-open-sound-control/
3. TCP, https://it.wikipedia.org/wiki/Transmission_Control_Protocol
4. TCP (Transmission Control Protocol): presentazione del protocollo di trasmissione, https://www.ionos.it/digitalguide/server/know-how/presentazione-tcp/
5. UDP, https://it.wikipedia.org/wiki/User_Datagram_Protocol