

Musikalischer Rahmen

Hochleistungs-Audio-Streaming-Framework für MOST

Die zunehmende Nachfrage nach Infotainment im Fahrzeug und die breite Anerkennung von MOST als Multimedia-Standard im Automobil haben den Weg für die Entwicklung eines kostengünstigen Streaming-Frameworks geebnet. Dieses Streaming-Framework ist eine strategische Lösung für diverse OEM-spezifische Anforderungen. Dieser Beitrag stellt das Design eines neuen Audio-Streaming-Frameworks für das MOST-Netzwerk vor.

Von Sabeen A. und Manju Venugopal

Die Nachfrage nach der Entwicklung einer Streaming-Plattform entsteht hauptsächlich aus den verschiedenen, von OEM-Herstellern aufgestellten Erfordernissen. Ein allgemeines Framework, das über verschiedene Plattformen hinweg leicht portierbar ist, bedeutet einen Vorteil für den Systementwickler.

Dieses allgemeine Framework konzentriert sich auf die Entwicklung von Streaming-Applikationen, die unabhängig von der NetServices-Schicht und der zu Grunde liegenden Plattform sind. Das Framework soll das Hinzufügen von verschiedenen Streaming-Formaten und die Nutzung mehrfacher Kanäle unterstützen – bei gleichzeitiger von den OEMs geforderter Flexibilität und einfacher Nutzung.

Bei Berücksichtigung der vorgeschlagenen Lösung können die oben genannten Anforderungen effektiv gehandhabt werden. Die folgenden Punkte sind die Hauptvorteile dieses Frameworks:

- ▶ Leicht konfigurierbar mit einem hohen Grad an Wiederverwendbarkeit.
- ▶ Leicht anpassungsfähig an verschiedene Plattformen.
- ▶ Unabhängig vom NSL.
- ▶ Unabhängig vom MOST-Geschwindigkeitsgrad.

- ▶ Effiziente Speicherausnutzung und wenig Rechenleistungsbedarf.
- ▶ MISRA-C-konform.

■ Herausforderung

In einem der Projekte wurde eine Streaming-Audio-Plattform von Drittanbietern verwendet. Diese Plattform half, das Prototypenmodell sehr leicht zu zeigen. Bei der Umsetzung des aktuellen Produkts traten aber verschiedene Probleme auf.

Das Hinzufügen der Unterstützung mehrerer Kanäle funktionierte mit der vorhandenen Plattform. Allerdings verringerte sich die Leistung und die Speicherauslastung nahm zu. Der nächste Schritt bestand darin, komprimierte Audio-Ströme zu unterstützen, aber das Framework war zu starr, so dass Teile des Codes umgeschrieben werden mussten. Der End-Quellcode war zu komplex und machte es schwierig, ihn weiterhin zu nutzen. Die zur Verfügung stehende Rechenleistung und der Speicher wurden fast völlig ausgenutzt. Letztendlich konnten keine neuen Funktionen hinzugefügt werden, ohne die gegenwärtige Leistung beträchtlich zu beeinflussen.

Deshalb wurde eine Streaming-Plattform erdacht, mit Fokus auf ein

leicht konfigurierbares Framework, um eine Unterstützung vielfacher Stream-Typen und Kanäle mit minimalen Anforderungen an Rechenleistung und Speicher hinzuzufügen. Der sekundäre Fokus sollte das Framework unabhängig von der zu Grunde liegenden NetServices-Schicht und den MOST-Geschwindigkeitsklassen machen.

Auf dem Markt sind teilweise implementierte Audio-Streaming-Plattformen verfügbar. Aber das Problem beim Einsatz solcher Plattformen ist, dass sie großen Speicherbedarf haben. Sie verursachen auch eine hohe Systemauslastung. Die hohen Kosten sind auch ein Grund, den Einsatz solcher Plattformen abzulehnen. Um einige der verfügbaren Plattformen zu verwenden, muss das Systemdesign angepasst werden, um die durch das Framework zur Verfügung gestellte Unterstützung zu erhöhen. Wenn die Systemanforderung darin besteht, mehrere Kanäle gleichzeitig zu unterstützen, sollte die vorhandene Plattform auf ein höheres Niveau skaliert werden. Die Skalierbarkeit ist im Falle der vorhandenen Plattformen nicht sehr leicht. Dieser Beitrag konzentriert sich auf die Herleitung einer Lösung für die oben angeführten Probleme, indem eine neue, einfache Plattform entwickelt wird, die in diversen Plattformen verwendet werden könnte, ohne dabei in großem Umfang Aspekte der Skalierbarkeit berücksichtigen zu müssen.

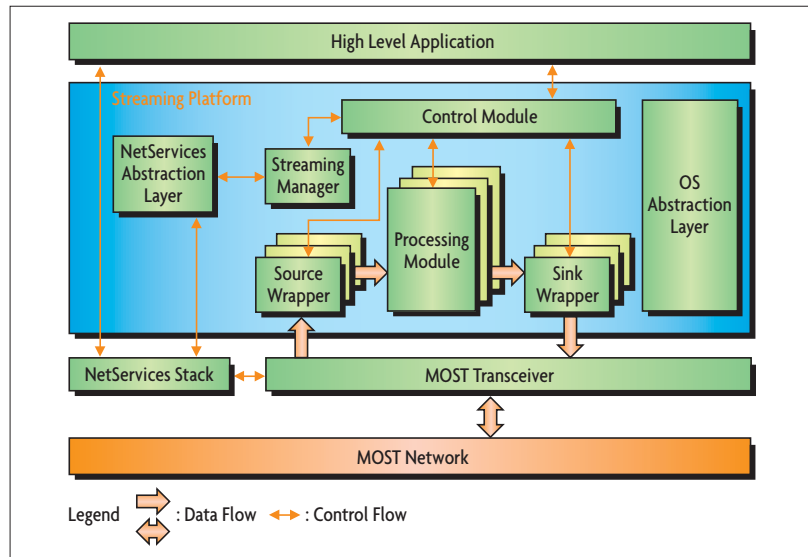
■ Architektur des Framework

Ein Hochleistungs-Infotainment-Framework im Fahrzeug ist erforderlich, um hohe Medienqualität mit geringer Verzögerung zu unterstützen. Das vorgeschlagene Framework kann für einen Slave-Knoten zusammen mit

Source- oder Sink-Geräten verwendet werden (Bild 1). Der Betriebssystem-Abstraktions-Layer erlaubt die leichte Migration zwischen verschiedenen Betriebssystemen. Das Framework wurde so entworfen, dass es in allen MOST-Systemen unabhängig von der Geschwindigkeitsklasse genutzt werden kann.

Die verschiedenen logischen Schichten werden basierend auf den ausgeführten Funktionen identifiziert. Die Streaming-Plattform mit Fokus auf kleinsten Speicherverbrauch wird spezifisch für embedded System-Applikationen entworfen. Das einfache Framework minimiert den Rechenaufwand und die Kopieroperationen, die gewöhnlich mit der Stream-Übertragung verbunden sind. Nachfolgend die Beschreibung der verschiedenen Module im System:

- ▶ **Steuereinheit:** Ressourcen-Management und Kontrolle der Kommunikation zwischen Streaming-Schicht und High-Level-Applikationen.
- ▶ **Streaming-Manager:** Dieses Modul beaufsichtigt die Streaming-Kanäle und bedient auch den Connection-Manager, der dem MOST-Transceiver zugeordnet ist.
- ▶ **Streaming-Schicht:** Eine skalierbare Streaming-Schicht wurde dahingehend entwickelt, Audio-Streaming zu unterstützen. Sie besteht aus dem Source-



! Bild 1. Architektur des High-Performance-Infotainment-Frameworks.

Wrapper, der den Audio-Input von mehreren Quellen bearbeitet, dem Rechenmodul, das die Audiosignalkette implementiert, und dem Sink-Wrapper, der ausgegebene Audiodaten an Lautsprecher, Kopfhörer oder zurück zum MOST-Transceiver routet.

▶ **NetServices-Abstraktions-Schicht:** Ein generischer Wrapper für die Kommunikation mit dem NetServices-Layer.

▶ **Betriebssystem-Abstraktions-Schicht:** Das Modul dient als Schnittstelle zu allen grundlegenden Funktionen des Betriebssystems.

Im Folgenden werden die Funktionen der einzelnen Schichten erläutert.

Steuereinheit

Die Steuereinheit kontrolliert den Funktions- und Datenfluss zwischen dem Source-Wrapper, dem Verarbeitungsmodul und dem Sink-Wrapper. Die Steuereinheit überwacht die Verbindungsrichtung zwischen der Source (den Sources) und der Sink (den Sinks). Dieses Modul empfängt Allokations-, De-Allokations-, Connect- und Disconnect-Anfragen von MOST

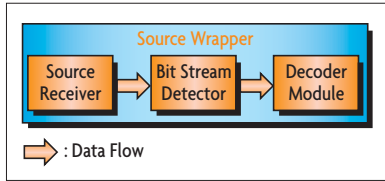


Bild 2. Der Source-Wrapper, als Teil der Streaming-Schicht, kann wiederum in mehrere Unterschichten aufgeteilt werden.

über die High-Level-Applikation und routet diese Nachrichten zum Streaming-Manager. Die Funktionen, die auf das Audio-Streaming bezogen, aber nicht auf die Befehle Play, Pause, Stopp-Signale etc. beschränkt sind, werden durch dieses Modul bearbeitet. Das Verhalten der Sink- und Source-Wrapper wird so durch die Steuereinheit geführt.

Streaming-Manager

Die Low-Level-Allokationen von Kanälen und Routing-Funktionen sind innerhalb des Network-Interface-Controllers zusammengefasst. Die Ressourcen des MOST-Transceivers werden vom Streaming-Manager verwaltet. Dieses Modul bearbeitet die Allokations- und De-Allokations-Anfragen des MOST-Knotens im Netzwerk, der den Streaming-Manager beaufsichtigt. Es berechnet wiederum die Kanalzuweisung und De-Allokations-Anfragen sowie die Verbindung zum und die Trennung vom Netz.

Streaming-Schicht

Der Grundaufbau der Streaming-Schicht besteht aus Source-Wrapper, Rechenmodul und Sink-Wrapper. Der Benutzer kann zur Unterstützung zahlreicher Kanäle der Audiodatenübertragung vielfache Instanzen der elementaren Schicht schaffen. Die Source-Wrapper-Schicht kann in verschiedene Unterschichten aufgeteilt werden (Bild 2):

- ▶ Source-Empfänger
- ▶ Bit-Stream-Detektor
- ▶ Decoder-Modul

Wie der Name andeutet, ist der Source-Empfänger befähigt, Daten von verschiedenen Quellen zu erhalten. Der Bit-Stream-Detektor beurteilt die eingehenden Daten und unterscheidet

die verschiedenen Bit-Ströme, d.h. AC3, DVD Audio, MPEG2 Audio, SACD und MPEG1/2 Audio. Das Signal wird dann zu den entsprechenden Decodern geleitet.

Die Datenausgabe des Decoder-Moduls wird zum Rechenmodul transportiert (Bild 3). Das Rechenmodul kann entsprechend der Systemanforderungen entworfen werden. Das Rechenmodul besteht aus verschiedenen Audio-Verarbeitungsblöcken, die die

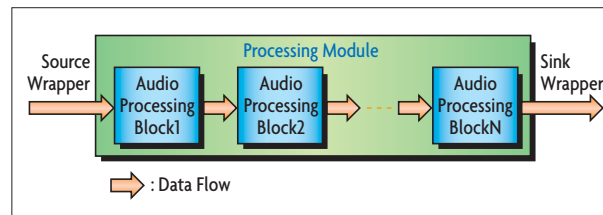


Bild 3. Die Recheneinheit (Processing Unit) kann flexibel an die Systemanforderungen angepasst werden.

erforderliche Signalkette implementieren. Die Scratch-Buffer müssen vom Benutzer für Audio-Verarbeitungserfordernisse entworfen werden. Die Scratch-Buffer-Allokation soll minimiert sein und ihren Schwerpunkt auf der Wiederverwendbarkeit aller Audio-Verarbeitungsblöcke haben. Der ganze Prozess kann vom Benutzer entworfen und nahtlos in das Streaming-Framework integriert werden.

Der Sink-Wrapper ist eine Schnittstellen-Schicht zwischen dem Rechenmodul und dem Sink-Gerät. Die durch das Rechenmodul modifizierten Daten werden zur Sink über die vom Sink-

Wrapper zur Verfügung gestellten Schnittstellen geroutet.

Die Streaming-Schicht verwertet Puffer für die Weiterleitung des Audio-Streams. Doppelte Puffer werden an Eingang und Ausgang der Source- und Sink-Wrapper verwendet. Für die Verarbeitung müssen doppelte Puffer der erforderlichen Größe im Framework reserviert werden. Bei einem typischen Beispiel eines 32-bit-Sample-Audio-Stream-Eingangs werden insgesamt drei Puffer benötigt, jeder der Größe $4 \times 32 \times 2$.

NetServices Abstraktions-Schicht

Der Netzwerk-Service-Stack

stellt APIs zur Verfügung, um die Streaming-Mechanismen zu vereinfachen. Das Framework sollte anpassungsfähig an die verschiedenen Versionen der auf dem Markt verfügbaren MOST-Stacks sein. Das wird durch den im Framework eingeschlossenen NetServices-Abstraktions-Layer realisiert.

Betriebssystem-Abstraktions-Schicht

Die Betriebssystem-Abstraktions-Layer zielt auf die Abstraktion von Betriebssystemfunktionen. Das erleichtert die Integration des Frameworks in verschiedene Plattformen. *sj*



Sabeen A.

arbeitet als Technischer Leiter für embedded Software-Produkte für das Unternehmen Network Systems and Technologies (P) Ltd. Er hat mehr als elf Jahre Berufserfahrung mit embedded Software und hat zum Design und der Implementierung verschiedener mobiler und automobiler Produkte beigetragen.
sabeen.a@nestgroup.net



Manju Venugopal

begann ihre Karriere bei Network Systems and Technologies (P) Ltd. Heute arbeitet sie als Software-Designerin bei Continental Automotive, Singapore. Ihr Abschluss in Electronics & Instrumentation Engineering von der Cochin University of Science & Technology war der zweitbeste ihres Jahrgangs.
itzmanju@gmail.com