Chapter 7

Multipoint Synchronization

7.1 Introduction

The real time applications like video conferencing, tele-teaching, telemedicine etc. are increasing day by day, resulting a drastic change in mode of communication in society and also increase the use of IP network. The real-time distributed multimedia systems are characterized by one or several sources transmitting (unicast or multicast) multimedia streams to one or several receivers, playing one or several of the streams at receiver side. Media can be categorized in static media and continuous media. Differing from static media; Continuous media (video and audio) have well-defined temporal relationships between subsequent Media Data Units (MDUs). After capturing media from different sources, it is digitized and packetized and send through the IP network from sender to receivers. Due to nondeterministic delay in IP network, jitter is introduced to the media packet, results change in order of subsequent media packet at receiver side. To maintain the order with temporal relationship between MDUs same as sender side, synchronization among the media packets at receiver side is necessary.

Multipoint Synchronization refers to the playout of media streams to all the receivers at the same time in multicasting scenario to present fairness to receivers. The example can be of tele-quiz application where the same video frame is needed to display at same
time to all the participating nodes. Other examples of such applications are Teleteaching, teleconferencing, e-meeting etc. There exist some protocols that are proposed to support such type of application. In this dissertation an algorithm is proposed to achieve multipoint multimedia synchronization.

### 7.2 Formal definition of problem statement

Let there are \( n \) no. of nodes communicating in multicasting scenario. A node can send up to \( m \) no. of streams.

\( N_i \) represent the \( i^{th} \) participating node where \( 1 \leq i \leq n \). \( S(N_i) \) represents node \( N_i \) is sender where \( 1 \leq i \leq n \). \( R(N_i) \) represents node \( N_i \) is receiver where \( 1 \leq i \leq n \). Taking the multicasting scenario, \( \exists N_i \ \forall N_j \ S(N_i) \rightarrow R(N_j) \) where \( 1 \leq i, j \leq n \) and \( i \neq j \). \( S_{ix} \) represents \( x^{th} \) media stream sent by node \( N_i \) where \( 1 \leq x \leq m \) and \( S(N_i) = \text{true} \). \( t_{ix}(p) \) represents time at which \( p^{th} \) frame of \( S_{ix} \) starts transmitting. \( d_{ijx}(p) \) represents delay introduced in \( p^{th} \) frame of \( S_{ix} \) at \( N_j \) where \( 1 \leq x \leq m \), \( S(N_i) = \text{true} \), \( R(N_j) = \text{true} \), \( 1 \leq i, j \leq n \) and \( i \neq j \). \( a_{ijx}(p) \) represents arrival time of \( p^{th} \) frame of \( S_{ix} \) at \( N_j \) where \( 1 \leq x \leq m \), \( S(N_i) = \text{true} \), \( R(N_j) = \text{true} \), \( R(N_k) = \text{true} \), \( 1 \leq i, j \leq n \) and \( i \neq j \).

Due to non deterministic delay in IP network, it may happens that \( \forall x \exists p \ d_{ijx}(p) \neq d_{ijx}(p) \) that implies \( \forall x \exists p \ a_{ijx}(p) \neq a_{ijx}(p) \) [arrival time = transmitting time + dealy in network] where \( S(N_i) = \text{true} \), \( R(N_j) = \text{true} \), \( R(N_k) = \text{true} \) and \( 1 \leq i, j, k \leq n \), \( i \neq j, i \neq k \). \( p_{ijx}(p) \) denotes presentation time of \( p^{th} \) frame of \( S_{ix} \) at \( N_j \) where \( 1 \leq x \leq m \), \( S(N_i) = \text{true} \), \( R(N_j) = \text{true} \) and \( 1 \leq i, j \leq n \), \( i \neq j \).

So, to find \( \text{adj}_{ijx}(p) \) = the adjusting time(for skipping or pausing) of \( p^{th} \) frame of \( S_{ix} \) at \( N_j \) where \( 1 \leq i \leq m \), \( S(N_i) = \text{true} \), \( R(N_j) = \text{true} \) and \( 1 \leq i, j \leq n \), \( i \neq j \); such that \( \forall p, x, j, k \ p_{t_{ijx}(p)} = p_{t_{ikx}(p)} \) where \( 1 \leq i, j, k \leq n \), \( i \neq j, i \neq k \) and \( S(N_i) = \text{true} \), \( R(N_i) = \text{true} \), \( R(N_k) = \text{true} \) and \( S_{ix}(p) = \text{true} \). \( R \) is a set that hold type of relation between MDUs of two different streams. A function \( \text{rel}(S_{ix}(p), S_{iy}(p)) \) maps relation of \( p^{th} \) frame of different stream \( S_{ix} \) and \( S_{iy} \) to set \( R \). If \( N_i \) where \( S(N_i) = \text{true} \), send more than one stream then it needs to find adjustment time.
such that $\forall \, p, \, x, \, j, \, k \, pt_{ijx}(p) = pt_{ikx}(p)$ where $1 \leq i, j, k \leq n, \, i \neq j, i \neq k, \, 1 \leq x \leq m$ and $S(N_i)=\text{true}, \, R(N_j)= \text{true}, \, R(N_k)= \text{true}$ and also $\forall p, \, x, \, y \, r(S_{ix}(p), \, S_{iy}(p)) \rightarrow R$ holds, where $1 \leq x, y \leq m \, \text{and} \, x \neq y$.

7.3 Metric definition

Two metrics are defined here to measure the performance of the synchronization algorithm with respect to loss of data and asynchrony among receivers.

7.3.1 Loss metric

Loss metric($M_l$) can measure the percentage of loss at receiver end with respect to the MDU received at receiver side.

Loss can be occurred due to network as well as the synchronization process. Let $R_i, \, N_i$ represents the no. of packet received and played for $i^{th}$ stream during time $t$ respectively. So, loss metric

$$M_l = \frac{\sum_{i=1}^{m}(R_i-N_i)}{\sum_{i=1}^{m}(R_i)} \times 100\%$$

7.3.2 Asynchrony metric

Two types of asynchrony metrics are defined here, relative asynchrony and overall asynchrony.

7.3.2.1 Relative asynchrony

The playout time difference of each packet for all media streams at one receiver with respect to playout time of the packet to another receiver is defined as the relative asynchrony between two receivers.
Let $P_{ij}$ represents no. of packet played out at receiver side in $i^{th}$ receiver for $j^{th}$ stream during time $t$. $P_{ij}(p)$ represents playout time of $p^{th}$ packet at receiver side in $i^{th}$ receiver for $j^{th}$ stream. Relative asynchrony between two receivers $i,k$ is measured by the equation:

$$\left| \sum_{i=1}^{n} \sum_{p=1}^{m} \frac{(P_{ij}(p) - P_{kj}(p))}{\sum_{p=1}^{m} (P_{ij})} \right| \forall i,j \text{ where } 1 \leq i, k \leq n \& i \neq k$$

7.3.2.2 Overall asynchrony

The playout time difference of each packet for all media streams at all receivers with respect to standard playout time (expected playout time of the packet calculated by some algorithm) of the packet in the system is defined as the overall asynchrony of the system. Standard playout time of packet $p$ in $j^{th}$ stream for $i^{th}$ receiver is $P_{std,ij}(p)$. So, overall asynchrony of system is -

$$\sqrt{\sum_{i=1}^{n} \sum_{p=1}^{m} \frac{(P_{std,ij}(p) - P_{ij}(p))}{\sum_{p=1}^{m} (P_{ij})} \forall i,j \text{ where } 1 \leq i, k \leq n \& i \neq k}$$

7.4 Intermedia specification

There is well defined temporal relationship between MDUs of continuous media as well as static media of different media streams. The well defined logical representation within MDUs of inter media is established according to their temporal relationship. The possible relationship of two MDUs may be within two continuous media or one continuous and one static media or both can be static. A specification is defined that can represent all possible relationship holds between two MDUs of different media streams. Size of MDU may or may not be same for two different media streams. Three types of relations can hold between MDUs - precedes, succeeds and overlaps. Relations are defined below using two MDUs of different media streams denoted as A and B. $ts(A)$ and $d(A)$ represent the starting time and duration of presentation of frame A.
7.4.1  A→pB

This relationship holds when A finishes its playout before starting the playout of B. Logical representation of the condition is \( ts(A) + d(A) \leq ts(B) \).

7.4.2  A→oB

This relationship holds when A and B satisfy one of the following three conditions. First condition:

A and B both start and finishes playout at the same time or A starts playout after the starting of B but finishes before the end of play out of B or A starts playout after starting of B but both finish together.

Second condition:

Frame A starts its playout after the start of B but ends after the end of B.

Third condition:

B starts its playout after the start of A but ends after the end of B.

Logical representation of the condition is

\[
( ts(A) = ts(b) \land ts(A) + d(A) = ts(B) + d(B) ) \lor ( ts(A) \geq ts(b) \land ts(B) + d(B) \geq ts(A) ) \lor ( ts(A) \leq ts(b) \land ts(A) + d(A) \geq ts(B) ).
\]

7.4.3  A→sB

This relationship holds if A starts its playout after the end of play out of B. Logical representation of the condition is \( ts(A) \geq ts(B) + d(B) \).

Table 7.1 represents all possible pictorial representations of inter media relationship and corresponding temporal relation.
Table 7.1: Intermedia specification

<table>
<thead>
<tr>
<th>Relation</th>
<th>pictorial representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A \rightarrow_p B$</td>
<td>![Diagram of A → p B]</td>
</tr>
<tr>
<td>$A \rightarrow_o B$</td>
<td>![Diagram of A → o B]</td>
</tr>
<tr>
<td>$A \rightarrow_s B$</td>
<td>![Diagram of A → s B]</td>
</tr>
</tbody>
</table>

### 7.5 Self modifying stochastic colour petri net

Self Modifying Stochastic Colour Petri Net is proposed as follows:

SMSCPN has 9 tuple \{P, T, A, λ, C, I, Tn, F, CL\}.

P: \{p_1, p_2, ..., p_x\} where \(x \geq 0\), is a finite set of places. \(P = P_N \cup P_F\) where \(P_N\) is the set of places where no function is executed in arrival of resource token and \(P_F\) is the set of places where some function is executed in arrival of resource token.

T: \{t_1, t_2, ..., t_m\} where \(m \geq 0\), is a finite set of transitions. \(T = T_I \cup T_T\) and \(T_I \cap T_T = \phi\) where, \(T_I\) is the set of immediate transitions that fire the token immediately when token is available at input place. \(T_T\) is the set of timed transitions that take some time to fire tokens from input place to output place.

A: \((P \times T) \cup (T \times P)\) is the finite set of arcs. \(A = I^- \cup I^+ \cup I^h\) and \((I^- \cup I^+) \cap I^h = \phi\) where, \(I^-\) refers finite set of input arcs and \(I^- \subseteq (P \times T)\). \(I^+\) refers finite set of output arcs and \(I^- = T \times P\). \(I^h\) refers finite set of inhibitor arcs i.e. if input place does not hold
any token then transition is enabled to fire and a token is produced in output place and \( I^h \subseteq P \times T \).

\( \lambda: \lambda_1, \lambda_2, .., \lambda_j \), where \( j \geq 0 \) and \( j \mid T \) is a finite set of transition rate assigned to timed transition.

\( C: \{c_1, c_2, ..., c_n\} \) where \( n \geq 0 \), is a finite sets of commands.

\( I: \{i_1, i_2, ..., i_k\} \) where \( k \geq 0 \), is a finite set of information flow through net.

\( T^n : \{(C^* - \varepsilon) \cup P(I)\} \), is a finite set of token. \( T^n = CT \cup RT \) and \( CT \cap RT = \phi \) where, \( CT = \{(C^* - \varepsilon)\} \) is a set of colour token. \( RT = P(I) \) is a set of resource token.

\( F: \{f_1, f_2, ..., f_l\} \) where \( l \geq 0 \) is a set of functions that execute in \( P_F \) when a resource token is arrived at the place. Function can generate colour token, modify the information in resource token, add new information to resource token i.e. change resource token.

\( CL: \) is the finite set of clock.

In the propose model places are represented by the circle, timed and immediate transitions are represented by the rectangle and bar, input and output arcs are represented by arrow, inhibitor arcs are represented by circle headed arrow, place, arc and transition created by colour token is represented by dashed line.

Set of commands with particular sequence represent by the colour token. It can change the net structure temporarily to accommodate with new environment and control the flow of resource token. Table 7.2 represents list of commands execute on different component of petri net in the model.

**Transition rules:** Let \( S=\{s_1, s_2, ..., s_n\} \) where \( n \geq 0 \), is the set of types of information(I). \( M: I \rightarrow S \) is a function that maps information to a type. Now \( \alpha: RT \rightarrow X \) where \( X = \{S^* - \varepsilon\} \), is a function where \( \forall rt \in RT, \alpha(rt) = x \) where \( x \in \{S^* - \varepsilon\} \) and if \( rt = \{i_1, i_2, ..., i_{|x|}\} \) then \( x = \{M(i_1), M(i_2), ..., M(i_{|x|})\} \). \( \beta: A \rightarrow \{S^* - \varepsilon\} \) is a function that binds arc to a type of resource token. A resource token can transit through a arc if \( \beta(x) = \alpha(rt) \) where \( a \in A \) and \( rt \in RT \).

\( \delta: CT \times RT \rightarrow CL \) is a function that maps colour token combined with resource token
Table 7.2: Command executed in SMSCPN

<table>
<thead>
<tr>
<th>Component of petri net</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Place</td>
<td>Create a place</td>
</tr>
<tr>
<td></td>
<td>delete a place</td>
</tr>
<tr>
<td>Transition</td>
<td>Enable transition</td>
</tr>
<tr>
<td></td>
<td>Disable transition</td>
</tr>
<tr>
<td></td>
<td>Create transition</td>
</tr>
<tr>
<td></td>
<td>Delete transition</td>
</tr>
<tr>
<td>Arc</td>
<td>Create a arc</td>
</tr>
<tr>
<td></td>
<td>delete a arc</td>
</tr>
<tr>
<td>Clock</td>
<td>Set the clock value</td>
</tr>
<tr>
<td></td>
<td>count down the clock</td>
</tr>
</tbody>
</table>

to a clock for timed transition. When clock value goes down to zero, the corresponding resource token transits to output place. When resource token transits from input place, colour token that is created at that place is deleted itself and also changes done by colour token is removed from net.

7.6 Multipoint multimedia synchronization model

In this section the scenario is modeled for three types of media streams, received at receiver node using the proposed SMSCPN tool. Here taking all possible case of arrival of media streams and show how to process these streams such that Multipoint Synchronization is achieved among receivers with inter stream synchronization at each node. There are some constrains to maintain real time interactive multimedia synchronization scenario. The maximum tolerable delay for interactive communications is 250ms, refer as primary delay constrain and acceptable asynchrony within audio video stream is 10ms. In the model control message is available at receiver side if the maximum delay between sender to all receivers is within primary delay constrain otherwise more QoS support is demanded for that receiver. Reference delay that is used to synchronize all receiver, calculated from that maximum delay value in control message. In the model I=\{audio, video, static media, network delay, time of arrival, duration of presentation, reference delay, generation time, waiting time\} and S=\{a, v, s, d, arr, du, dref, ger, wt\}. Different resource token used in
this model are of the following types.

\[ x_1 = (a, d, \text{arr}, \text{du}); \ x_2 = (\text{dref}); \ x_3 = (x_1, x_2); \ x_4 = (x_3, \text{wt}); \ x_5 = (v, d, \text{arr}, \text{du}); \ x_6 = (x_5, x_2); \ x_7 = (x_5, \text{wt}); \ x_8 = (x_1, x_5); \ x_9 = (s, d, \text{arr}, \text{du}); \ x_{10} = (x_9, x_2); \ x_{11} = (x_9, \text{wt}); \ x_{12} = (x_1, x_9). \]

### 7.6.1 Multipoint synchronization model for multiple media streams

In Figure 7.1 the model for *Multipoint Synchronization* is shown for one audio, video and static media. It consists of 25 places and 36 transitions. Here \( P_F = \{ p_2, p_3, p_4, p_5, p_6, p_{10}, p_{11}, p_{18}, p_{17}, p_{16}, p_{15}, p_{13}, p_{14}, p_{21}, p_{22}, p_{23}, p_{24} \} \) and \( P_N = \{ p_1, p_8, p_7, p_9, p_{12}, p_{19}, p_{20}, p_{25} \} \). Transition \( t_1 \) transits at arrival rate of audio frame receive from internet. Place \( p_1 \) acts as the receiving buffer. When token is available at \( p_1 \) and control message is available at \( p_7 \) then resource token is taken one by one for processing in place \( p_2 \). Hence inhibitor arc is used at transition \( t_2 \). Function \( f_1 \) executed at place \( p_2 \) is:

```plaintext
while resource token arrives do
    if delay of MDU \leq \text{reference delay} then
        create colour token \( ct_1 \)
    else
        create colour token \( ct_2 \)
    end if
end while
```

\( ct_1 \) executes the following commands

1. Disable transition \( t_4 \)
2. Enable transition \( t_3 \)

\( ct_2 \) executes the following commands

1. Disable transition \( t_3 \)
2. Enable transition \( t_4 \)

If transition \( t_3 \) fires then resource token transits to \( p_3 \). Function \( f_2 \) executed at \( p_3 \) is:

```plaintext
while resource token arrives do
```

125
calculate waiting time of MDU
Create resource token of type x4

end while

If transition \( t_4 \) fires then resource token transits to \( p_4 \). Function \( f_3 \) executed at \( p_4 \) is-

\[
\text{while resource token arrives do}
\]

\[
\text{if delay of MDU} \leq \text{reference delay} + 10\text{ms} \leq 250\text{ms then}
\]

\[
\text{calculate waiting time of MDU}
\]

\[
\text{Create resource token of type x}4
\]

\[
\text{else}
\]

\[
\text{create colour token ct}_3
\]

\[
\text{end if}
\]

\[
\text{end while}
\]

Colour token \( \text{ct}_3 \) executes the commands-

1. Disable transition \( t_6 \)

2. Create place \( p_8 \)

3. Create transition \( t_9 \)

4. Create arc \( p_4t_9, t_9p_8 \)

When \( t_5 \) or \( t_6 \) is fired resource token transits to \( p_5 \) and \( p_7 \) according to type bound to the arc. Function \( f_4 \) executed at place \( p_5 \) is-

\[
\text{while resource token arrives do}
\]

\[
\text{create colour token ct}_4
\]

\[
\text{end while}
\]

Colour token \( \text{ct}_4 \) executes following commands-

1. Set a clock to the transition with information of waiting time associate with resource token.
Figure 7.1: multipoint synchronization model
2. Count down the clock.
After firing of \( t_7 \) token transits to place \( p_6 \) and function \( f_5 \) executed at \( p_6 \) is-

\[
\textbf{while} \text{ resource token arrives do} \\
\text{create colour token } ct_5 \\
\textbf{end while}
\]

Colour token \( ct_5 \) executes the following commands-

1. Set the clock to transition \( t_8 \) with fixed value of frame duration.

2. Count down the clock.
When transition \( t_8 \) is fired then token leaves the net that means the media frame is played out synchronously at receiver side.

Transition \( t_{10} \) transits at arrival rate of packet received from internet. Place \( p_9 \) stores the resource token coming from network. Now video stream is synchronized at multipoint in absence of audio stream otherwise it is synchronize with the audio stream. According to normal transition rule either transition \( t_{11} \) or \( t_{12} \) fires. Transition \( t_{11} \) is fired if audio is not available at \( p_1 \) and resource token transit to \( p_{10} \). Function \( f_6 \) executed at \( p_{10} \) is-

\[
\textbf{while} \text{ resource token arrives do} \\
\text{if } \text{delay of MDU} \leq \text{reference delay} + 30\text{ms} \text{ then} \\
\text{calculate waiting time of MDU} \\
\text{Create resource token of type } x7 \\
\text{else} \\
\text{create colour token } ct_6 \\
\text{end if} \\
\textbf{end while}
\]

Colour token \( ct_6 \) executes some commands-

1. Disable \( t_{13} \)

2. Create place \( p_{12} \)
3. Create transition $t_{14}$

4. Create arc $t_{10}p_{14}$ and $t_{14}p_{12}$

When transition $t_{14}$ is fired, the token transits to new created place $p_{12}$, that means system demands new control message for continuing the communication. When $t_{13}$ is fired resource token transits to $p_{13}$ and $p_7$ according to type bound to the arc. If $t_{12}$ is fired in presence of audio frame at $p_1$ video resource token transit to place $p_{11}$ when audio resource token come to place $p_2$ for processing. Function $f_7$ executed at place $p_{11}$ is-

```
while resource token arrives do
    if token holds succeeds relation with audio token then
        create colour token $c_{10}$
    else if token holds precedes relation with audio token then
        create colour token $c_8$
    else
        create colour token $c_{12}$
    end if
end while
```

Colour token $c_8$ executes some commands-

1. Disable $t_{17}$ and $t_{18}$

2. Enable $t_{19}$

Colour token $c_{10}$ execute commands-

1. Disable $t_{19}$ and $t_{17}$

2. Create place $p_{17}$

3. Create arc $t_{18}p_{17}$

4. Enable $t_{18}$

Colour token $c_{12}$ executes commands-

1. Disable $t_{19}$ and $t_{18}$
2. Enable $t_{17}$

According to the colour token created at place $p_{11}$ transition $t_{19}$ or $t_{17}$ or $t_{18}$ is fired and token transit to $p_{18}$ or $p_{15}$ or $p_{16}$ and $p_{17}$ respectively. At $p_{18}$ function $f_8$ is executed.

\[\textbf{while} \text{ resource token arrives do} \]
\[\text{create colour token } ct_9 \]
\[\textbf{end while} \]

Colour token $ct_9$ executes following commands-

1. Create transition $t_{23}$ and $t_{24}$.

2. Create an inhibitor arc $p_2t_{24}$ and normal arc $p_7t_{24}$, $p_2t_{23}$, $t_{24}p_{10}$, $t_{23}p_{11}$, $p_{18}t_{24}$, $p_{18}t_{23}$.

If $t_{23}$ is enable then token again compared with next audio token otherwise $t_4$ is enable and token transit to place $p_{10}$ and process accordingly. Function $f_9$ executed at place $p_{17}$ is-

\[\textbf{while} \text{ resource token arrives do} \]
\[\text{create colour token } ct_{11} \]
\[\textbf{end while} \]

Colour token $ct_{11}$ performs following commands

1. Create transition $t_{22}$

2. Create arc $p_9t_{22}$, $t_{22}p_{11}$, $p_{17}t_{22}$.

After firing of $t_{22}$ token transits to $p_{11}$. Function $f_{10}$ executed at $p_{15}$ and $p_{16}$ is-

\[\textbf{while} \text{ resource token arrives do} \]
\[\text{calculate waiting time of MDU} \]
\[\text{Create resource token of type } x_7 \]
\[\textbf{end while} \]

When $t_{13}$ or $t_{20}$ or $t_{21}$ is fired token moves to place $p_{13}$. Function $f_4$ is executed at that place. After firing of $t_{15}$, the token moves to place $p_{14}$ and $f_{11}$ is executed given
while resource token arrives do
    create colour token ct_7
end while

Colour token ct_7 executes following commands-

1. Enable transition t_{25}.

2. Set the clock to transition t_{16} with value of frame duration.

3. Count down the clock.

When t_{16} is fired resource token transit to place p_{19} and block until t_{25} is enable.

Transition t_{26} is fired if static media data is arrived from Internet. Place p_{20} is stored the resource token coming from network. Now this media data can be synchronized at multipoint in absence of audio stream otherwise it is synchronized with the audio stream. Transition t_{27} is fired if token is available at p_{1}. After occurrence of t_{27}, resource token transits to p_{21}. f_{12} is executed at p_{21}.

while resource token arrives do
    calculate waiting time of MDU
    Create resource token of type x11
end while

When t_{29} occurs, token moves to p_{7} and p_{23} according to type bound with arc. If t_{28} is fired, resource token transit to place p_{22} when audio resource token come to place p_{2} for processing. At p_{22} function f_{13} is executed.

while resource token arrives do
    if token holds succeeds relation with audio token then
        create colour token ct_{13}
    else if token holds precedes relation with audio token then
        create colour token ct_{15}
    else
        create colour token ct_{14}
end while
end if  
end while

Colour token \( ct_{13} \) executes following commands-

1. Create transition \( t_{32} \) and \( t_{33} \)

2. Create a inhibitor arc \( p_{2}t_{32} \) and normal arc \( p_{2}t_{33}, p_{7}t_{32}, t_{31}p_{21}, t_{32}p_{22}, p_{22}t_{32}, P_{22}t_{33} \).

Colour token \( ct_{14} \) executes few commands-

1. Disable transition \( t_{30} \)

2. Enable transition \( t_{31} \)

Colour token \( ct_{15} \) executes few commands-

1. Disable transition \( t_{31} \)

2. Enable transition \( t_{30} \)

After execution of commands by colour token if transition \( t_{13} \) is fired the token transit to place \( p_{25} \). At that place \( f_{12} \) is executed. When \( t_{29} \) or \( t_{35} \) is fired resource token transits to \( p_{23} \) and \( f_{4} \) is executed. When \( t_{34} \) is fired resource token transits to \( p_{24} \) and \( f_{5} \) is executed at that place. When \( t_{36} \) is fired token leaves the net.

### 7.6.2 Probabilistic analysis of the model

Using self modifying stochastic colour petri net the stochastic nature of the system can be modeled. The conflict can be resolved between two transitions by imposing probability to the transitions. In this model for audio, let token is arrived maintaining the Poisson process at an average rate \( \alpha \) then in time interval \( t \) probability that there is a token in place \( p_{1} \) is \( P_{p_{1}}=\sum_{k=1}^{\infty} \frac{\alpha^{k}e^{\alpha t}}{k!} \). As delay in network layer follows the normal distribution with mean and variance \( \mu \) and \( \sigma \) respectively, control message available at \( p_{7} \) i.e. there is a token at \( p_{7} \) is \( P_{p_{7}}=\int_{d_{min}}^{250} \frac{e^{-\frac{(x-\mu)^{2}}{2\sigma^{2}}}}{\sqrt{2\pi\sigma^{2}}} dx \). Rate of transition \( t_{2} \) is 1 but it is marking dependent. So the probability that there is a token at \( p_{2} \) is same as probability of \( (P_{p_{1}}, P_{p_{7}}) \). Then there is a conflict between \( t_{3} \) and \( t_{4} \). Now \( t_{3} \) is enabled when network delay
incurred to the token is less than the reference delay. So probability of enabling transition t\text{3} is \( x = \int_{d_{\text{min}}}^{d_{\text{ref}}} e^\frac{-(x-\mu)^2}{2\sigma^2} \, dx \). Now transition t\text{4} occurs when delay is larger than ref delay i.e. the probability of enabling transition t\text{4} is \( y = \int_{d_{\text{ref}}}^{\infty} e^\frac{-(x-\mu)^2}{2\sigma^2} \, dx \). So there is a token in p\text{3} with probability \( P_{P3} = (P_{P2} \cdot x) \). Rate of transition t\text{5} is 1, only depend on presence of token in p\text{3}. Similarly the Probability that there is a token at p\text{4} is \( P_{P4} = P_{P2} \cdot y \).

Here probability of enabling the transition t\text{6} is \( z = \int_{d_{\text{ref}}}^{d_{\text{ref}+10}} e^\frac{-(x-\mu)^2}{2\sigma^2} \, dx \) and probability of enabling transition t\text{9} is \( \int_{d_{\text{ref}+10}}^{\infty} e^\frac{-(x-\mu)^2}{2\sigma^2} \, dx \). So probability that there is a token at place p\text{5} is \( P_{P5} = (P_{P3} + P_{P4} \cdot z) \). Network delay distribution for particular source and destination in IP network follows the normal distribution. Here the play out time is shaped according to a fixed reference delay for a session (until the reference delay information is changed). So the time for waiting before presentation is also maintained normal distribution with mean and variable \( \mu_1 \) and \( \sigma_1 \) respectively. So the mean time of transition t\text{6} is \( tt_1 = \int_{0}^{d_{\text{ref}+10-d_{\text{min}}}} e^\frac{-(x-\mu_1)^2}{2\sigma^2} \, dx \). So rate of transition at t\text{6} is \( R_1 = \frac{P_{P6}}{tt_1} \). So probability of presence of a token at place p\text{6} is \( P_{P6} = e^{-R_1 t} \). At place p\text{6} token is blocked for fixed time that equals to the time is taken by an audio MDU to generate at sender side. If generation rate at sender side is R2 then the final rate at which the token is departed i.e. the rate of transition t\text{8} is \( P_{P6} \cdot R_2 \). Also taking measure of arrival rate at t\text{1} and departure rate at t\text{8}, buffer size can be calculated such that no loss occurs due to shortage of buffer.

In the same manner the video stream can be analyzed. For same receiver let token arrives from network at rate \( \beta \) i.e. transition rate of t\text{10} is \( \beta \). As arrival process of data stream is followed Poisson process so p\text{9} hold at least one token within t time with probability \( P_{P9} = \sum_{k=1}^{\infty} \frac{\beta^k e^{R_1 t}}{k!} \). Now transition t\text{11} or t\text{12} is fired depending on the resource token available in place p\text{1}. So probability that a token is in place p\text{10} is \( P_{P10} = P_{P9} \cdot (1-P_{P1}) \cdot P_{P7} \). Now transition t\text{13} occurs if delay information carried by token is within reference delay addition with 30ms. So probability within this range is \( a = \int_{d_{\text{min}}}^{d_{\text{ref}+30}} e^\frac{-(x-\mu)^2}{2\sigma^2} \, dx \) and and t\text{14} is fired with probability \( b = \int_{d_{\text{ref}+30}}^{\infty} e^\frac{-(x-\mu)^2}{2\sigma^2} \, dx \). The system is failed with probability \( P_{P10} \cdot b \). Now t\text{12} is fired when audio master stream available. So probability of enabling the transition t\text{12} is \( (P_{P1} \cdot P_{P7} \cdot P_{P9}) \). For inter stream synchronization there is no need any delay information for finding the presentation time of MDU. Here delay
can be in any range. As no token is going to place p_{12} from place p_{11} probability that a
token in place p_{13} is P_{p_{13}} = (P_{p_{10}} \cdot a + P_{p_{11}}).
likewise audio the time for waiting before
presentation for video is also maintained normal distribution with mean and variable \mu^2
and \sigma^2 respectively. The token can block for mean time \int_{d_{ref} + 30}^{d_{min}} e^{-\frac{(x-\mu^2)^2}{2\sigma^2}} x dx.
So rate of transition t_{15} is R3 = (P_{p_{13}}/tt_2). So probability of a token in place P_{p_{14}} is e^{-R3t}.
If mean and variance of presentation time is \mu^3 and \sigma^3 then the mean delay of transition
t_{16} is \int_{0}^{30} e^{-\frac{(x-\mu^3)^2}{2\sigma^3}} x dx. So rate of transition t_{16} is R4 = P_{p_{14}}/tt_3. Probability of
presence of a token at place P_{p_{14}} is e^{-R4t}. Now transition t_{25} is enabled with probability
of (1 - e^{-R4t}) and it is an immediate transition. So final rate of transition t_{25} is P_{p_{14}}(1-
\text{e}^{-R4t}). Buffer size can be measured from this analysis similarly as audio stream.
Static media can also be analysed in the same way as of video frames.

7.7 Algorithm for Multipoint Synchronization

According to the proposed model for Multipoint Synchronization two things are consid-
ered, maintaining the quality of presentation of media stream with minimum significant
data loss in Multipoint Synchronization process and using dynamic delay information to
synchronize the multipoint as mean delay between sender and receivers vary with time.

7.7.1 Initial and periodic synchronization

For doing the clock synchronization initially and at regular interval NTP algorithm is
used at the server end and NTP date at the receiver end. The NTP service is provided
by the network server located at internet. To resolve the asynchrony between different
receivers during the session skipping or pausing are used but not in periodical manner.
Skipping is done in the silent part so to avoid significant data loss.
### 7.7.2 Delay and expected playout time calculation

In this algorithm to calculate the expected playout time at receiver end two parameters are needed, one is the maximum delay at that moment among all delay between sender and receiver and another is extra minimum buffering time use for de-jittering. A delay matrix is given in Table 7.3 store the most updated information about network delay between all nodes in a scenario. The information of delay matrix is used to decide whether a node can able to continue in conferencing scenario or not. If the delay between sender and receiver cross the delay constrain then the receiver cannot continue with conferencing scenario and should take necessary step for increasing the quality of service in network layer.

Consider the scenario of an audio conferencing session consisting of n nodes, represented by $N_1$, $N_2$, ..., $N_n$ where any of the n nodes may act as the sender. For any sender, all the receivers have realized separate amount of network delay that can vary with time and store in delay matrix. Each row of the matrix represents the amount of delay between a sender to all receivers. $D_{ij}$ represents the delay between the sender $N_i$ to receiver $N_j$. Whenever the sender sends a data packet, the entries of the corresponding row is continuously updated. $D_{imin} = \min(D_{ij})$ for all $j=1,2,..n$ from $N_i$ and $D_{imax} = \max(D_{ij})$ for all $j=1,2,..n$ from $N_i$. Now assume the source starts the transmission at time $t_0$. Network delay can cause the receiver to start its play out process as early as $t_0+D_{imin}$ or as late as $t_0+D_{imax}$ that causes an initial asynchrony of at most $D_{imax}-D_{imin}$ between each pair of receivers. Let $\rho$ be the drift in the playout rate. Let $\Delta$ be the additional time added to allow the initial MDUs of each stream to arrive and be buffered at all receivers. $\Delta=(D_{imax}-D_{imin})/(1-\rho)$ Then if $t_0$ is the time at which the packet is sent, then the expected playout instant of each of the packet is calculated as $T_{expected}=t_0+D_{imin}+\Delta$.

<table>
<thead>
<tr>
<th></th>
<th>$N_1$</th>
<th>..</th>
<th>$N_j$</th>
<th>..</th>
<th>$N_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_1$</td>
<td>X</td>
<td>..</td>
<td>$D_{1j}$</td>
<td>..</td>
<td>$D_{1n}$</td>
</tr>
<tr>
<td>$N_i$</td>
<td>$D_{i1}$</td>
<td>..</td>
<td>$x$</td>
<td>..</td>
<td>$D_{in}$</td>
</tr>
<tr>
<td>$N_n$</td>
<td>$D_{n1}$</td>
<td>..</td>
<td>$D_{nj}$</td>
<td>..</td>
<td>$x$</td>
</tr>
</tbody>
</table>
where $T_{\text{expected}}$ is the time represents the time at which the packet has to be played and reference delay is equals to $(D_{\text{imin}} + \triangle)$.

### 7.7.3 Synchronous playout algorithm

Synchronous play out algorithm calculates the time that a frame must wait before presentation and the playout time of the frame using the expected playout time calculation. Stream that is synchronized at multipoint, known as master stream and streams that is synchronized at multipoint according to master stream, known as slave stream and this mechanism of synchronization is called master-slave mechanism. This mechanism is used in the algorithm to achieve Multipoint Synchronization for multimedia in a multicasting scenario. For different streams there is different receiving buffer for receiving the data from Internet. Each receiver follows the Algorithm 1.

A thread is started for each slave stream during execution of Algorithm 1 that process the MDU of slave stream using intermedia specification between them given in Algorithm 2.

All the functions call from Algorithm 1 and Algorithm 2 use following notation:

- Delay information of audio frame $(\text{delay})$,
- Time of arrival of audio frame $(\text{arr})$,
- Reference delay of the system $(\text{dref})$,
- Time of arrival of video frame $(\text{arr})$,
- Waiting time of the frame of master stream $(\text{wt}_m)$,
- Arrival time difference between video frame and corresponding fame of master stream $(\text{atd})$,
- Generation time difference of video frame and corresponding fame of master stream $(\text{std})$,
Algorithm 1: synchronous playout algorithm

while conference is going on do
    if audio stream is available then
        Select any one of audio stream as master stream.
        Take audio MDU of master stream from buffer
        process-audio-frame
        for each different stream available do
            Start a thread to process MDU of slave stream
        end for
    else if video stream is available then
        Select video stream as master stream.
        Take video MDU of master stream from buffer
        Call process-video-frame
        for each different media available do
            Start threads to process MDU of slave streams
        end for
    else
        Select any static media as master stream.
        Take MDU of master stream from buffer
        Call process-static-frame
        for each different media available do
            Start a thread to process MDU of slave stream
        end for
    end if
end while
Algorithm 2 Thread to process MDU of slave stream

if MDU precedes MDU of master stream then
    while MDU precedes MDU of master stream = true do
        if stream is audio then
            Call process-audio-frame
        else if stream is video then
            Call process-video-frame
        else
            Call process-static-frame
        end if
        Take next MDU
    end while
else if MDU overlaps MDU of master stream = true then
    if stream is audio then
        Call process-audio-slave
    else if stream is video then
        Call process-video-slave
    else
        Call process-static-slave
    end if
else
    return to the buffer
end if
Duration of video frame (du).

The information produce from the function is

Presentation time(pt) and Playout duration(du).

---

**Algorithm 3** process-audio-frame

**Require:** flag ← 0 and temp ← 0
diff←(dref-delay)

if diff≥0 and flag=0 then
    wt ← diff
else if diff≤-10 and flag=0 then
    wt ← 0, flag ← 1,temp ← −diff
else if diff≥0 and flag=1 then
    wt ← (temp + diff)
else
    if (temp+diff)≥0 then
        wt ← (temp + diff)
    else
        if diff≤-10 then
            wt ← 0, temp←−diff
        else
            system failed
        end if
    end if
end if
pt←(arr + wt)

---

### 7.8 Result and discussion

Taking a scenario where three receivers were receiving audio data and mean delay was different for different receivers. Delay was generated randomly for each packet. Simulated the scenario using the proposed algorithm and also without the proposed algorithm (only each receiver use the buffer for de-jitter mechanism) and plotted the playout time against frame number. When presentation time of a frame becomes zero it specifies packet loss.
Algorithm 4 process-video-frame

\[
\text{diff} \leftarrow (\text{dref}-\text{delay})
\]

\[
\text{if } \text{diff} \geq 0 \text{ then}
\]

\[
\text{wt} \leftarrow \text{diff}
\]

\[
\text{else if } \text{diff} \leq -30 \text{ then}
\]

\[
\text{wt} \leftarrow 0, \text{du} \leftarrow (\text{du} - \text{diff})
\]

\[
\text{else}
\]

\[
\text{system failed}
\]

end if

\[
\text{pt} \leftarrow (\text{arr} + \text{wt})
\]

Algorithm 5 process-static-frame

\[
\text{diff} \leftarrow (\text{dref}-\text{delay})
\]

\[
\text{if } \text{diff} \geq 0 \text{ then}
\]

\[
\text{wt} \leftarrow \text{diff}
\]

\[
\text{else}
\]

\[
\text{wt} \leftarrow 0, \text{du} \leftarrow (\text{du} - \text{diff})
\]

end if

\[
\text{pt} \leftarrow (\text{arr} + \text{wt})
\]

It is clear from Figure 7.2 that there are seven packet loss in simulation without using the algorithm and presentation time distribution for audio stream is different in different receivers. According to loss metric, for this scenario \(\text{Ml} = \frac{7}{60} \times 100\% = 11\%\). It is noticed from Figure 7.3 that the presentation time distribution for audio stream in different receivers is merged into one line and there is no loss due to synchronization process while simulating the same scenario with the algorithm. In scenario 1 the 1st and 2nd receiver are taken and calculated the relative asynchrony using equation given in section 5. Taking the playout time of frame given by simulation (without using the proposed algorithm) relative asynchrony between receiver1 and receiver2 is equals to 52.87ms which is not less than 10 ms whereas using the proposed algorithm it becomes 0.345ms which is almost zero.

Taking another scenario where two receivers were receiving two continuous media (one audio and another video with different frame size) with different mean delay for different receivers. Delay is generated randomly for each packet. In Figure 7.4 the presentation time is plotted against frame number for both receivers. The generation time is also
Algorithm 6 process-audio-slave

Require: flag ← 0 and temp ← 0

diff←(\(w_{m}-\text{std}+\text{atd}\))

if diff≥0 and flag=0 then
  wt ← diff
else if diff≤-10 and flag=0 then
  wt ← 0, flag ← 1, temp ← -diff
else if diff≥0 and flag=1 then
  wt ← (temp + diff)
else
  if (temp+diff)≥0 then
    wt ← (temp + diff)
  else
    if diff≤-10 then
      wt ← 0, temp ← -diff
    else
      wt ← 0, du ← 0
  end if
end if

pt←(arr + wt)

Algorithm 7 process-video-slave

diff←(\(w_{m}-\text{std}+\text{atd}\))

if diff≥0 then
  wt ← diff
else if diff≤-30 then
  wt ← 0, du ← (du − diff)
else
  wt ← 0, du ← 0
end if

pt←(arr + wt)
Algorithm 8 process-static-slave

diff←(wt\_m-std+atd)
if diff≥0 then
    wt ← diff
else
    wt ← 0, du ← (du - diff)
end if
pt←(arr + wt)

Figure 7.2: Graph for scenario1 without using algorithm
Figure 7.3: Graph for scenario 1 using algorithm

Figure 7.4: Graph for scenario 2
plotted to compare the inter relationship between two media streams. From the Figure 7.4 shows the presentation time of two different media streams at different receiver is same and also the vertical difference between two different media streams is same as the generation time which proofs that the synchronization process at multipoint does not changes the inter media relationship. In the second scenario, taking the calculated play out time given by simulation the overall asynchrony is calculated using the metric given in section 7.3. Overall asynchrony value is come in following manner- 11ms, 2.14 ms, 3.152 ms, 0ms, 6.43ms in 5 consecutive simulation of second scenario. All the value is very close to zero. So to conclude that the proposed algorithm is succeeded to solve the problem.

7.9 Conclusion

An algorithm is developed that works in application layer to resolve the Multipoint Synchronization issues in real time multimedia communication. Also some metrics are defined that verify the algorithm and got satisfactory result regarding Multipoint Synchronization and packet loss due to synchronization process.