Chapter 6

Interactive Group Synchronization

6.1 Introduction

Interactive Group Synchronization can be seen as multiple users synchronization. This synchronization task aims at avoiding semantics incoherence in the dialogue between all users. This situation happens, when a large number of distributed nodes participating in a conference session and more than one audio streams generate from more than one nodes at the same time and reach the mix streams to the listener’s nodes, it become a non understandable audio. So audio stream generated at different nodes must be synchronized with each other. Petri net provides useful mathematical formalism for modeling concurrent system and their behaviors. An example of High level Petri Nets is Coloured Petri Nets, which is a graph oriented language for design, specification, simulation and verification of systems. It is in particular well-suited for systems that consist of a number of processes which communicate and synchronize. Its a combination of Petri Nets and programming language where Petri Nets control the structures, synchronization, communication and resource sharing and data manipulation are described by functional programming language. With this background an Interactive Group Synchronization algorithm (IGS Algorithm) is proposed and modeled through time coloured Petri nets and are explained with an example.
6.2 Modeling the problem

6.2.1 Time Coloured Petri Nets

Coloured Petri Nets is a graph oriented language for design, specification, simulation and verification of systems. It is in particular well-suited for systems that consist of a number of processes which communicate and synchronize; its a combination of Petri Nets and programming language. Where Petri Nets control the structures, synchronization, communication and resource sharing, where as data manipulation are described by functional programming language.

It constructs of:

- Ellipses and circles: are called places, they describe the states of the system.
- Rectangles: are called transitions, describe the action.
- Arrows: are called arcs, arc expression described how the state of the CP-net changes when transitions occur.
- Tokens: Each place contains a set of tokens, each of token carries a data value with a given type called colour.

![Diagram of Coloured Petri Nets](image)

Figure 6.1: Example of Coloured Petri Nets

In figure [6.1] transition SendPacket is enabled when there is a token \((n, p)\) on place Send, where \(n = \text{packetno.}\) and \(p = \text{packetcontent.}\) When transition occurs token is
removed from input places to place $A$ (input buffer). Transition $Transmit\ Packet$ is enabled when $< n, p, OK = true >$, then packet is transferred from place $A$ to $B$. Transition $Received\ Packet$ is enabled when token is on place $B$, packet $Received$ at Place $Received$ and packet is $OK$. A CPN associated with a time interval is called $Time\ Coloured\ Petri\ Nets$ (TCPN), shown in figure 6.2. It consists of two types of places one Time Place, another Non-time Place and a Transition. Time Place: Associated with time interval. When a token is created in a time place the token is locked for a duration of fixed time interval. Token becomes unlocked at the output arc when the duration of time interval over. Non-time Place: A token created in non-time place is to be unlocked at all times. Transition: Transition is enabled if all of its input places contain at least one unlocked token.

![Figure 6.2: TCPN](image)

### 6.2.1.1 Formal definition of TCPN

A $TCPN$ is a tuple $TCPN = (\Sigma, P, T, A, N, C, I, O, t)$

Where:
(i) Σ is a finite set of non-empty types, called colour sets.
(ii) P = (p₁, p₂, p₃, ......... pₘ), where m ≥ 0 is a finite set of places.
(iii) T = (T₁, T₂, T₃, ......... Tₙ), where n ≥ 0 is a finite set of transition.
where P ∩ T = Ø, i.e.: set of places and transition are disjoint.
(iv) A is finite set of arcs such that: P ∩ T = P ∩ A = T ∩ A = Ø.
(v) N is a node function. It is defined from A into P x T ∪ T x P.
(vi) C is a colour function. It is defined from P into Σ.
(vii) I: P → T, is input arc mapping from place to transition.
(viii) O: T → P, is output arc mapping from transition to place.
(ix) t = (t₁, t₂, t₃, ..... t₉), where q ≥ 0, is a finite set of time interval.

6.2.1.2 Properties of TCPN Model

6.2.1.2.1 Safety Property This property states that the teacher terminate the lecture session when all the listeners are allowed to speak. This property is represented by ∀x : x ∈ N, Terminated(t) → Accepted(x).

6.2.1.2.2 Liveness Property This property states that every transition in the TCPN model is firable from every states of the reachability graph in a finite time.

6.2.1.2.3 Reachability Property The speech must be reachable to the every participant of the lecture session. After a firing sequence token must be reached from initial marking to final marking means all the transitions are successful.

6.2.2 TCPN Model of the problem

In a tele lecture session, there is a coordinator, who controls the entire session. At first Coordinator behaves as a sender, starts the conference with his speech and the others behave as receiver, and they receive the speech and make request or question against the
particular speech. Now in this modeling two behaviors are described, Sender behaviors: who gives the speech and Receiver Behavior: who listens to the speech and make the request or question against the sender speech.

![Figure 6.3: Sender Behavior](image)

In figure 6.3 initially the Send place has only one token with coordinator $S_{ID}$ and Set $S_{ID}$ place binding the Coordinator $S_{ID}$ and Speech ($Sp$). Transition send speech is enabled, because there is a $S_{ID}$ token on place Send and ($S_{ID}, Sp$) token on place Set $S_{ID}$. When transition occurs it removes the two specified token from the input places, but the token $S_{ID}$ immediately put back on place Send, due to double arc. So coordinator speech sent by adding it to the input buffer($A$) of the network. When the ($S_{ID}, Sp$) token is put on place $A$, transition Transmit speech become enabled with two different condition if $OK(S, r) = True$ and $OK(S, r) = False$ where $S = Set$ of $Speech – ID$ of a particular conference session and the function $OK(S, r)$ checks whether $r \in S$ for $r \in (S_{ID} – 1, S_{ID} – 2, ......S_{ID} – n)$. If $OK(S, r) = True$ then speech transferred from place $A$ to $B$ otherwise discard. When ($S_{ID}, Sp$) token arrives at place $B$ then transition Receive speech becomes enabled.

In the receiver side, only those receivers, who have a request or question against the speech are only considered as a receiver. In the above figure 6.4 after transition Received Speech is enabled every receiver binding with their $RQ_{ID}$ and Request Speech and copy
Figure 6.4: Receiver Behavior

to the place $C_1,C_2,\ldots,C_n$ respectively where $n$ is the number of receiver who have a request or question. When $(RQ,ID,Sp)$ token is at place $C$ then transition Transmit speech works as a similar way to Transmit speech of Sender behavior and request moves from place $C$ to $D$. Place $D$ is Request STACK buffer where all the requests are stored at this place with their speech. Request STACK buffer operates with condition that if STACK is Empty then STOP otherwise POP one element from STACK and place on the output arc of place $D$.

In figure 6.5 when token request ID and speech$(RQ,ID,Sp)$ reach at time place then it locked for $2.5d$ (here $2d$ is considered as round trip delay between sender and receiver in the network and $0.5d$ is considered as request processing time) time duration and then the token is unlocked at input arc of transition, now transition is enabled with unlocked token $(RQ,ID,Sp)$ and it is transferred to the input arc of Non-time place.

At Set ID place set the new $S,ID$ ($S,ID = RQ,ID$) for next Speaker. These entire step continues until the Request STACK buffer is empty.
Now combining these three behaviors which make a complete TCPN Model for the above problem shown below in figure 6.6

**Declaration:**

```
type INT = integer;
type DATA = Audio packet;
type BOOL = Boolean;
type INT × DATA = product INT*DATA;
var S_ID : INT;
var Sp : DATA;
var ok : BOOL;
```
6.3 Formal Verification of the TCPN Model

Different state diagrams of the TCPN model are presented for its verification. In state diagram at first teacher behave as a speaker to start the lecture session with his speech and others behave as listener who receive the speech and make request against the speakers speech. Now taking \( x \) and \( y \) as any participant of lecture session where \( x, y \in N \), where \( N \) is a set of participant in the Lecture session. In a particular time instance \( x \) behave as a speaker and \( y \) as a listener or vice verse.

In the state diagram when \( x \) is the speaker then \( x \) delivers the speech to the \( y \), where \( y \) is a set of listener and \( x \cup y = \text{Participants} \). It can be described by

\[
\forall y \exists x \text{Sp}(y, x) \Rightarrow \text{TRUE}
\]

Where \(|x| = 1\) and \( x, y \in \text{Participants} \)

\( \text{Sp}(y, x) : x \text{ delivering speech to } y \)

After delivering the speech, \( x(\text{speaker}) \) waits for request from \( y(\text{listener}) \)

After listening the speech when listener makes request to the speaker that can be described by
\[ \exists x, \exists y \ RSp(x, y) \rightarrow TRUE \]

where \(|x| = |y| = 1.

\( RSp(x, y) \): \( y \) makes request to \( x \).

All request against the particular speech that can be described by

\[ \forall y, \exists x \ S\!p(x, y) \rightarrow TRUE \]

where \(|y| \geq 0, |x| = 1.

\( S\!p(x, y) \): where \( y \) is the listener who sends the request to the speaker \( x \). After receiving all the request, a participant corresponding to the request ID on the top of the \( RSB \) is selected as speaker, that can be described by \( \exists x_1 \ S\!p_s(x_1) \rightarrow TRUE \) where \( x_1 \) is Listener and \( \text{Rqid}(x_1) = RSB[\text{TOP}], \text{Rqid}(x) \rightarrow \text{Request ID of } x \), is a new speaker.

When all the listeners get chance to give the speech then Request STACK buffer \( (RSB) \) generate the final speech sequence at input arc of \( \text{Accepted} \) place that can be describe by \( \exists x S\!p_s(x) \rightarrow TRUE \), where \( 1 \leq x \leq n \), \( n \) is number of participants in the lecture session.

When \( S\!p_s(x) \) in the input arc of \( \text{Accepted} \) place then pending of listener request is end and \( S\!p_s(x) \) forwarded to the output arc of \( \text{Accepted} \) place then the lecture session terminated.

### 6.3.1 Speaker state diagram of TCPN Model

![Figure 6.7: Speaker State Diagram of TCPN Model](image)

A teacher \( (t) \) is a initiator of a lecture session so at first \( t \) is the speaker in the speaker state diagram in figure 6.7.

(i) When speech is not yet delivered, the state **Quiet** holds.
(ii) When speaker is waiting for listeners request the state **waiting** holds.
(iii) When speaker receive the entire request against his speech then state **terminated** holds.

Speaker t always remain one of the three states, denoted by 

\[(\forall t, t \in N) \text{ Quiet}(t) + \text{ waiting}(t) + \text{ terminated}(t) = 1 \] 

where \( N \) is set of participants and the above expression is a Boolean expression.

**6.3.2 Listener state diagram of TCPN Model**

![Listener state diagram of TCPN Model](image)

In the listener state diagram in figure 6.8,

(i) When listener is not getting any speech from the speaker the state **uninformed** holds.
(ii) When the listener is waiting to give the speech, the state **pending** holds.
(iii) When all the listener gets chance to deliver speech then the state **Accepted** holds.

Listener x remains in one of the three state denoted by 

\[(\forall x, x \in N) \text{ Uninformed}(x) + \text{ Pending}(x) + \text{ Accepted}(x) = 1 \] 

The above expression is a Boolean expression.
Now combining the speaker and listener state diagram of TCPN model, the combined model is shown in figure [6.9]

![State Diagram of TCPN Model]

Figure 6.9: State Diagram of TCPN Model

Now considering speaker and listener behavior of figure [6.9]  
(∀x, ∀y, x, y ∈ N)  
Quiet(y) + Uninformed(y) + Pending(x,y) + Pending(y,x) + terminated(x) + Accepted(x) = 1........(3)

Where (i)Final speech sequence is in Request STACK buffer then RSB holds.  
(ii)When y make a request to x but not getting chance to speak then Pending(x,y) holds. Similarly for a request from x to y then Pending(y,x) holds. To show that the teacher terminates the session when all other participants request are accepted, it is sufficient to show that a teacher only terminates when no participants request is uninformed or pending.  
So to prove,  
Terminated(t) → Accepted(x).
6.3.3 Lemma 1

TCPN model of the problem satisfies the Safety property.

Proof: From the equation (3)
\[(\forall x, \forall y : x, y \in N) : \text{Pending}(x, y) \rightarrow (\neg \text{Uninformed}(y)) \land \neg \text{accepted}(y))........(4)\]

Equation (2) implies
\[(\forall y, y \in N) (\neg \text{Uninformed}(y) \land \neg \text{accepted}(y)) \rightarrow \text{pending}(y)........(5)\]

Now combining (4) and (5) together, it is found
\[(\forall x, \forall y : (x, y) \in N \text{Pending}(x, y) \rightarrow (\text{pending}(y) \land y = t))......(6)\]

Where \(y = t\) means teacher is the initiator of lecture session and \(\text{pending}(y)\rightarrow\) Listener \(y\) is in pending state.

For each participant \(x\) there is another participant \(y\), who is the parent of \(x\). If \(x\) is pending then a sequence of pending state can be obtained. Therefore it can be written as
\[(\forall x, x \in N) : \text{Pending}(x) \rightarrow \exists z : \text{Pending}(z, t)........(7)\]

Where \(z\) is all pending student and \(x \subset z\).

Equation (3) implies
\[\text{Pending}(z, t) \rightarrow \neg \text{terminated}(t)........(8)\]

Combining equation (7) and (8) together,
\[\text{Pending}(x) \rightarrow \neg \text{terminated}(t)........(9)\]

From equation (3)
\[\text{Uninformed}(x) \rightarrow \neg \text{Accepted}(x)........(10)\]

When listener send request to the teacher and they are in pending state then \(\text{Pending}(t, x)\) holds where \(x\) is the listener in pending state and in the end the teacher must terminate the session.

Therefore
\[(\forall x, x \in N) : \text{Pending}(x) \rightarrow \exists t : \text{Pending}(t, x)........(11)\]

Which means \(x\) is waiting for \(t\) to give him chance to speak. From equation (3)
\[\text{Pending}(x) \rightarrow \neg \text{Uninformed}(x) \land \neg \text{Accepted}(x)........(12)\]

\[\neg \text{Uninformed}(x) \land \neg \text{Accepted}(x) \rightarrow \text{pending}(x)........(13)\]

So, from equation (9) and (13),
\[\neg \text{Uninformed}(x) \land \neg \text{Accepted}(x) \rightarrow \text{terminated}(t)\]
So, *Terminated (t) → Accepted(x)*.

### 6.3.4 Lemma 2

**TCPN model of the problem satisfies the Reachability and liveness property.**

The components of TCPN Model shown in figure 6.6 are designated as follows for proving the reachability of the model.

In transition state diagram of figure 6.10 Place Set *S* is described as *p*1, *send* place as *p*2, *S* place as *p*3, *A* place as *p*4, *B* place (*B*1, *B*2,...,*B*n) as *p*5, *C* place (*C*1, *C*2,...,*C*n) as *p*6, *D* place as *p*7 and received place as *p*8 and transition from *send* place to *A* place as *t*1, set *s* id place to *A* as *t*1, *S* place to *B* place as *t*2, *A* place to *B* place as *t*2, *B* place to *C* place as *t*3, *C* place to *D* place as *t*4. *D* place to set *S* ID place as *t*5, *D* place to receive placed as *t*5 and a transition *t*6 from received place that consumes the received token.

1) Transition state diagram of TCPN Model:

In the reachability graph shown in figure 6.11 a sequence of transition *σ* = *t*1, *t*2, *t*3, *t*4, *t*5 is a firing sequence from *μ*0

\[ μ0 \rightarrow^{t1} μ1 \rightarrow^{t2} μ2 \rightarrow^{t3} μ3 \rightarrow^{t4} μ4 \rightarrow^{t5} μ5 \]

we also write \( μ0 \rightarrow^{σ} μn \)

Now P and T represent place and transition respectively where P=(*p*1,*p*2,...,*p*k) and T=(*t*1,*t*2,...,*t*m).

We define k×m incidence matrix [T] where \([T](i,j) = φ(t_i,p_i) - φ(p_i,t_j)\)

\(φ(t_j,p_i) = \text{no. of token added.}\)

\(φ(p_i,t_j) = \text{no. of token remove.}\)

\([T](i,j) = \text{Changed in place } i \text{ when transition } j \text{ fires once.}\)

Now if marking is reachable then equation \(μ0 + [T]. # σ = μ \) holds. Where \(μ0 = \text{Initial marking, } μ = \text{Final marking, } # = m \text{ dimensional vector with its } j^{th} \text{ entry denoting}\)
the no. of time transition $t_j$ occurs in $\sigma$ and $[T]=\text{Incidence matrix}$.

So, $\mu_0 + [T] \cdot \# \sigma = \mu$ holds for firing sequence $\sigma = t_1, t_2, t_3, t_4, t_5$. So the tokens are reached to the final marking by sequence of successful transition. In the reachability graph, it shows that from every state, all the transitions $(t_1, t_2, t_3, t_4, t_5, t_6)$ are fireable in finite time. Therefore the TCPN model satisfies liveness property.
6.4 Proposed IGS Algorithm

6.4.1 Terminology Used

Conference Session: A *conference session* is defined as the duration for which a conference proceeds. The coordinator (defined later) starts and ends the session. A session consists of a coordinator and a number of participants (defined later) who exchange real time video communication and are connected through network in a distributed environment.

Participant: A *participant* is defined as a node which is connected in the conference session. A participant normally watches the conference (both audio & video) and also may participate through a speech or question. For this purpose, it has to inform the coordinator through a request. A participant can deliver his speech only on getting the consent from the coordinator.

Coordinator: *Coordinator* is one of the participants who controls the entire session. In a teacher-student environment, the teacher normally plays the role of the coordinator. All the requests generated by the participants in time scale are stored in first come first serve
basis but processed by the coordinator (also called Group Server) so that all the participants get the opportunity to deliver the speech or ask the question in an appropriate
sequence. The term appropriate sequence means that all the speech related to a specific context should be discussed in sequence (see algorithm). The coordinator can stop any participant at any point of time and may choose any other instead.

Request Tree: It is a tree of participant, whose root node is T, which symbolizes, that the coordinator has initiated the session. In any level of the tree the children represent that they have something to speak with the context of the speech of their parent node.

Request-Id: Whenever a participant sends a request to speak out with the context of someones speech, a unique request-id will be generated for that particular request. From the format of the request-Id, it can be identified, the position of the request, i.e. on whose context the participant would like to speak.

Speech-Id: When a participant is allowed to speak depending on his request, a unique speech-id will be generated for that participants speech.

6.4.2 Basic idea of the algorithm

This algorithm works in the group server. It chooses one participant at a time in a particular pattern of session topic to act as a speaker and other as listener to avoid the semantics incoherence in the dialogue between more than one participant. With this idea every participant in the group gets the turn to act as speaker. This continues for the entire life time of the conference session. For a participant when it gets its turn for speaking, either he can speak out or he can discard the chance and pass it to the next participant for acting as speaker. At the beginning of the session, each node sends a dummy packet with time stamp to all other nodes for finding the round trip delay (Drt). With this it creates a matrix $Drt[n][n]$ where $Drt[i][j]$, denotes the round trip delay between the $i^{th}$ node and $j^{th}$ node and the maximum network delay $d$ is defined as: $d = (\max(Drt[i][j]))/2$, for $i = 1$ to $n$ and $j = 1$ to $n$. It may also be said that after sending packet by any participant, it will take maximum $d$ time to reach all other participants.
6.4.3 Algorithmic steps

This section discuss the distributed algorithm for serving the request of the participants. The objective of the algorithm is to organize the conference discussion, by ordering all the participants from the request tree in the appropriate sequence.

Algorithm: IGS Algorithm.

Input:
i. Initiation of the conference session.
ii. Request by the participant to speak out.

Output:
i. Set of speech id.
ii. Set of request id.
iii. Request tree.

C1. Coordinator:
1. Initiates the conference session.
2. Generate the speech-id (1 in this case).
3. Transmit the speech-id to all connected in the session.
4. Deliver the speech (assume coordinator is the first speaker).
5. On completion of the speech, declare speech end.

P1. Participant:
1. Listen for the speech from the coordinator.
2. If interested, note the speech-id and accordingly raise request, wait for instruction from coordinator.
3. Else do nothing.

C2. Coordinator:
1. Wait for 2.5 d for receiving requests from participant(s).
2. If no request received, declare end of session, stop.
3. Else generate request-id and make request tree by inserting request-id as the child node of the previous speaker.
4. Explore the request tree in DFS order and transmit the request-id of first un-explored node to its corresponding participant (Pi).
5. Generate the speech-id for Pi and transmit it to all.
6. Wait for more requests

**P2. Participant:**
1. Pi: Receive the requestid and speak out and declare end of speech.
2. Other: Listen to the speech from Pi, if interested, note the speech-id and accordingly raise request, wait for instruction from coordinator.

**C3. Coordinator:**
1. If more requests reach, extend the tree and explore the tree else explore the tree for unexplored node.
2. If unexplored node found transmits the request-id of it to the corresponding participant else declare end of session and stop.
3. Go to step C2.5.

**Complexity:** For \( n \) number of participants in the conference session, there can be maximum \((n - 1)\) number of child in each node in the Request tree. Therefore it is an \( n \)-ary tree. The worst case complexity of the algorithm will be \( O(n^n) \). But to limit the discussion upto the depth of 10 in the request tree, the complexity will be \( O(n^{10}) \).

### 6.4.4 Illustration with example

**Request Stack Buffer Operation** in TCPN Model request STACK buffer (RSB) works as follow:

In figure [6.13](#) first the co-ordinator (T) initiate the conference session by giving his
first speech. On his speech, there are some queries from the participants s3, s5 and s1. So, the request from s3, s5 and s1 have arrived and are pushed in the stack of RSB. To form the request tree s3, s5 and s1 are put as the child node of T. With the pop operation on RSB, a request (s1) from the stack is popped out and s1 is allowed to give speech.

After the speech of s1, there are queries from s7 and s4 in figure 6.14. So, request from s7 and s4 are put as the child node of s1 in the request tree. Also, s7 and s4 are pushed in the stack of RSB. With the pop operation on RSB, a request s4 from stack is popped out and s4 is allowed to give his speech.

After the speech of s4, in figure 6.15, there are no queries against the speech of s4. So, the request s7 is popped out with the pop operation on RSB and s7 is allowed to give his speech.
After the speech of $s_7$, in figure 6.16, there are no queries against the speech of $s_7$. So, the request $s_5$ is popped out with the pop operation on $RSB$ and $s_5$ is allowed to give his speech.
After the speech of $s_5$, there are queries from $s_9$ and $s_8$ in figure 6.17 on the context of $s_5$. So, request from $s_9$ and $s_8$ are put as the child node of $s_5$ in the request tree. Also, $s_9$ and $s_8$ are pushed in the stack of $RSB$. With the pop operation on $RSB$, a request $s_8$ from stack is popped out and $s_8$ is allowed to give his speech.

![Figure 6.18: Step 6 of RSB & Request Tree Operation](image)

After the speech of $s_8$, in figure 6.18, there is no query against the speech of $s_8$. So, the request $s_9$ is popped out with the pop operation on $RSB$ and $s_9$ is allowed to give his speech.

![Figure 6.19: Step 7 of RSB & Request Tree Operation](image)

After the speech of $s_9$, there are queries from $s_{10}$ and $s_2$ in figure 6.19 on the context of $s_9$. So, request from $s_{10}$ and $s_2$ are put as the child node of $s_9$ in the request tree. Also, $s_{10}$ and $s_2$ are pushed in the stack of $RSB$. With the pop operation on $RSB$, a request $s_2$ from stack is popped out and $s_2$ is allowed to give his speech.

After the speech of $s_2$, in figure 6.20, there is no query against the speech of $s_2$. So,
the request s10 is popped out with the pop operation on RSB and s10 is allowed to give his speech.

After the speech of s10, in figure 6.21 there is no query against the speech of s10. So, the request s3 is popped out with the pop operation on RSB and s3 is allowed to give his speech.

After the speech of s3, there is no query against speech of s3. So, the stack is empty, which denotes the end of session.

**Final Speech Sequence** Final Sequence with Coordinator Speech is as follows in figure 6.22 which proves that the proposed synchronization algorithm can reduce the audio speech overlapping by avoiding the multiple speakers to speak at the same time. The algorithm chooses one participant at a time as a speaker and the other as listener so as to complete the session in a finite time.
A new type of synchronization called *Interactive Group Synchronization* has been addressed and a solution to achieve appropriate ordering has been presented. Such synchronization is very important in distributed video conferencing based Teleteaching system in order to ensure that all sender sends and receive their audio stream synchronously. Here an extension of petri nets called TCPN is defined to model the problem and formally verified its correctness by proving reachability, liveness and safety property of the model.