Summary

- Petri-Net Models
  - Definitions
  - Modeling protocols using Petri-Nets
  - Modeling Queueing Systems using Petri-Nets
- Max-Plus Algebra
Marked Petri Net Graph

- A Petri net graph is a weighted bipartite graph \( PN = (P, T, A, w, x) \)
- \( P \) is a finite set of places, \( P = \{p_1, \ldots, p_n\} \)
- \( T \) is a finite set of transitions, \( T = \{t_1, \ldots, t_m\} \)
- \( A \) is the set of arcs from places to transitions and from transitions to places
  - \( (p_i, t_j) \) or \( (t_j, p_i) \) represent the arcs
- \( w \) is the weight function on arcs
- \( x \) is the marking vector \( x = [x_1, \ldots, x_n] \) represents the number of tokens in each place.

Petri Net Example

- \( I(t_j) = \{p_i \in P : (p_i, t_j) \in A\} \)
- \( O(t_j) = \{p_i \in P : (t_j, p_i) \in A\} \)
- \( J(p_i) = \{t_j \in T : (t_j, p_i) \in A\} \)
- \( O(p_i) = \{t_j \in T : (p_i, t_j) \in A\} \)
Petri Net Marking

- A transition \( t_j \in T \) is **enabled** when each input place has at least a number of tokens equal to the weight of the arc, i.e.,
  \[ x_i \geq w(p_i, t_j) \text{ for all } p_i \in I(t_j) \]

- When a transition fires it removes a number of tokens (equal to the weight of each input arc) from each input place and deposits a number of tokens (equal to the weight of each output arc) to each output place.

![Petri Net Diagram](image)

Petri Net Dynamics

- The state transition function \( f \) of a Petri net is defined for transition \( t_j \) if and only if
  \[ x_i \geq w(p_i, t_j) \text{ for all } p_i \in I(t_j) \]

- If \( f(x, t_j) \) is defined, then we set
  \[ x'_i = x_i + w(t_j, p_i) - w(p_i, t_j) \text{ for all } i = 1, \ldots, n \]

- Define
  - \( u_j = [0, \ldots, 0, 1, 0, \ldots, 0] \) where all elements are 0 except the \( j \)-th one.
  - Also define the matrix \( A = [a_{ij}] \) where
    \[ a_{ij} = w(p_i, t_j) - w(p_j, t_j) \]

- In vector form
  \[ x' = x + u_j A \]
Example: Computer Basic Functions

- Design a Petri-net that imitates the basic behavior of a computer, i.e., being down, idle or busy

![Petri-net diagram]

Modeling Protocols Using Petri Nets (Stop and Wait Protocol)

- The transmitter sends a frame and stops waiting for an acknowledgement from the receiver (ACK)
- Once the receiver correctly receives the expected packet, it sends an acknowledgement to let the transmitter send the next frame.
- When the transmitter does not receive an ACK within a specified period of time (timer) then it retransmits the packet.
Stop and Wait Protocol: Deadlock

Avoidance Using Timers
Stop and Wait Protocol: Loss of Acknowledgements

Transmitter State

Channel State

Receiver State

Example: Queueing Model

What are some possible Petri nets that can model the simple FIFO queue

\[
x' = x + u \begin{bmatrix} 1 \\ -1 \end{bmatrix}
\]

\[
x' = x + u \begin{bmatrix} 1 & 0 \\ -1 & 0 \end{bmatrix} \quad x_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}
\]
Example: Queueing Model

A “richer” Petri-net model

\[ x' = x + u \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & -1 \\ 0 & -1 & 1 \end{bmatrix} \]

\[ x_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \]

Example: Queueing Model with Server Breakdown

How does the previous model change if server may break down when it serves a customer?

\[ x' = x + u \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & -1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \]

\[ x_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \]
Example: Finite Capacity Queueing Model

What is a Petri net model for a finite capacity queue?

\[ p_4 = K \]

\[ t_1 = a \]

\[ p_1 = Q \]

\[ t_2 = s \]

\[ p_2 = B \]

\[ t_3 = d \]

\[ x' = x + u \]

\[
\begin{bmatrix}
1 & 0 & 0 & -1 \\
-1 & 1 & -1 & 0 \\
0 & -1 & 1 & 1
\end{bmatrix}
\]

\[ x_0 = [0 \ 0 \ 1 \ K] \]

Other Petri Net Variations

- Inhibitor Arcs: A transition with an inhibitor arc is enabled when
  - All input places connected to normal arcs (arrows) have a number of tokens at least equal to the weight of the arcs and
  - All input places connected to inhibitor arcs (circles) have no tokens.

\[ p_1 \]

\[ t \]

\[ p_2 \]

\[ \text{DISABLED} \]

\[ p_1 \]

\[ t \]

\[ p_2 \]

\[ \text{ENABLED} \]

\[ p_1 \]

\[ t \]

\[ p_2 \]

\[ \text{DISABLED} \]
Other Petri Net Variations

- Colored Petri Nets
  - In this case, tokens have various properties associated with them. This can be an attribute or an entire data structure. For example,
    - Priority
    - Class
    - Etc.

Timed Petri Net Graph

- In the previous discussion, the Petri net models had no time dimension. In other words, we did not consider the time when a transition occurred.

\[ PN = (P, T, A, w, x, V) \]

- Timed Petri nets are similar to Petri nets with the addition of a clock structure associated with each timed transition
- A timed transition \( t_j \) (denoted by a rectangle) once it becomes enabled fires after a delay \( v_{jk} \).
Example: Timed Petri Net

Transitions $t_1$ and $t_3$ fire after a delay given by the model clock structure.
Transition $t_2$ fires immediately after it becomes enabled.

Petri Net Timing Dynamics

- **Notation**
  - $x$ is the current state
  - $e$ is the transition that caused the Petri net into state $x$
  - $t$ is the time that the corresponding event occurred
  - $e'$ is the next transition to fire (firing transition)
  - $t'$ is the next time the transition fires
  - $x'$ is the next state given by $x' = f(x, e')$.
  - $N_i$ is the next score of transition $i$
  - $y_i'$ is the next clock value of transition $i$ (after $e'$ occurs)
The Event Timing Dynamics

- **Step 1**: Given \( x \) evaluate which transitions are enabled
- **Step 2**: From the clock value \( y_i \) of all enabled transitions (denoted by \( \Gamma(x) \)) determine the minimum clock value
  \[ y^* = \min_{i \in \Gamma(x)} \{ y_i \} \]
- **Step 3**: Determine the firing transition
  \[ e' = \arg \min_{i \in \Gamma(x)} \{ y_i \} \]
- **Step 4**: Determine the next state
  \[ x' = f(x, e') \]
  where \( f() \) is the state transition function.

The Event Timing Dynamics

- **Step 5**: Determine
  \[ t' = t + y^* \]
- **Step 6**: Determine the new clock values
  \[ y'_i = \begin{cases} 
  y_i - y^* & \text{if } i \neq e' \text{ and } i \in \Gamma(x) \\
  v_{i,N_i+1} & \text{if } i = e' \text{ or } i \notin \Gamma(x) 
  \end{cases}, \quad i \in \Gamma(x') \]
- **Step 7**: Determine the new transition scores
  \[ N'_i = \begin{cases} 
  N_i + 1 & \text{if } i = e' \text{ or } i \notin \Gamma(x) \\
  N_i & \text{Otherwise}
  \end{cases}, \quad i \in \Gamma(x') \]
Two Operation (Dioid) Algebras

- The operation of timed automata or timed Petri nets can be captured with two simple operations:
  - **Addition** \( a \oplus b \equiv \max \{a, b\} \)
  - **Multiplication** \( a \otimes b \equiv a + b \)
- This is also called the **max-plus algebra**

---

Basic Properties of Max-Plus Algebra

- **Commutativity**
  \[
  a \oplus b = \max \{a, b\} = \max \{b, a\} = b \oplus a \\
  a \otimes b = a + b = b + a = b \otimes a
  \]
- **Associativity:**
  \[
  (a \oplus b) \oplus c = \max \{\max \{a, b\}, c\} = \max \{a, \max \{b, c\}\} = a \oplus (b \oplus c) \\
  (a \otimes b) \otimes c = a (b \otimes c)
  \]
- **Distribution of addition over multiplication**
  \[
  (a \oplus b) \otimes c = \max \{a, b\} + c = \max \{a + c, b + c\} = (a \otimes c) \oplus (b \otimes c)
  \]
- **Null element**
  \[
  a \oplus \eta = a \quad a \otimes \eta = \eta \quad \text{For example let } \eta = -\infty
  \]
Example

\[
\begin{bmatrix}
1 & 0 \\
2 & -2
\end{bmatrix} \otimes \begin{bmatrix}
2 & -1 \\
3 & 1
\end{bmatrix} = \begin{bmatrix}
1 & 0 \\
2 & -2
\end{bmatrix} \begin{bmatrix}
2 & -1 \\
3 & 1
\end{bmatrix}
\]

\[
= \begin{bmatrix}
\max\{1+2,0+3\} & \max\{1-1,0+1\} \\
\max\{2+2,-2+3\} & \max\{2-1,-2+1\}
\end{bmatrix}
\]

\[
= \begin{bmatrix}
3 & 1 \\
4 & 1
\end{bmatrix}
\]

\[a \begin{bmatrix}
3 & 1 \\
4 & 1
\end{bmatrix} = \begin{bmatrix} a + 3 & a + 1 \\
\end{bmatrix}
\]

Queueing Models and Max-Plus Algebra

\[a_k = a_{k-1} + v_{a,k}\]

\[d_k = \max\{a_k, d_{k-1}\} + v_{d,k}\]
Queueing Dynamics

- Let $a_k$ be the arrival time of the $k$-th customer and $d_k$ its departure time, $k=1,...,K$, then
  \[ a_k = a_{k-1} + v_{a,k} \]
  \[ d_k = \max\{a_k, d_{k-1}\} + v_{d,k} \]
  \[ = \max\{a_{k-1} + v_{a,k}, d_{k-1}\} + v_{d,k} \quad k=1,2,..., a_0=0, d_0=0 \]

- In matrix form, let $x_k = \begin{bmatrix} a_k & d_k \end{bmatrix}^T$ then
  \[
  x_{k+1} = \begin{bmatrix}
  v_{a,k} & -L \\
  v_{a,k} + v_{d,k+1} & v_{d,k+1}
  \end{bmatrix}
  \begin{bmatrix}
  a_k \\
  d_k
  \end{bmatrix}
  = A_k x_k \quad x_0 = \begin{bmatrix} 0 \\
  0 \end{bmatrix}
  \]

  where $-L$ is sufficiently small such that $\max\{a_k + v_{a,k}, d_k - L\} = a_k + v_{a,k}$

Example

- Determine the sample path of the FIFO queue when
  \[ v_a = \{0.5, 0.5, 1.0, 0.5, 2.0, 0.5, \ldots\} \]
  \[ v_d = \{1.0, 1.5, 0.5, 0.5, 1.0, \ldots\} \]

  \[
  x_{k+1} = \begin{bmatrix} a_{k+1} \\
  d_{k+1}\end{bmatrix} = \begin{bmatrix}
  v_{a,k} + a_k \\
  \max\{a_k + v_{a,k}, d_k\} + v_{d,k+1}
  \end{bmatrix} \quad x_0 = \begin{bmatrix} 0 \\
  0 \end{bmatrix}
  \]

  \[
  x_1 = \begin{bmatrix} a_1 \\
  d_1\end{bmatrix} = \begin{bmatrix}
  v_{a,1} + a_0 \\
  \max\{a_0 + v_{a,1}, d_0\} + v_{d,1}
  \end{bmatrix} = \begin{bmatrix} 0 + 0.5 \\
  \max\{0 + 0.5, 0\} + 1 \end{bmatrix} = \begin{bmatrix} 0.5 \\
  1.5 \end{bmatrix}
  \]

  \[
  x_2 = \begin{bmatrix} 0.5 + 0.5 \\
  \max\{1, 1.5\} + 1.5 \end{bmatrix} = \begin{bmatrix} 1 \\
  3 \end{bmatrix} \quad x_3 = \begin{bmatrix} 1.0 + 1.0 \\
  \max\{2, 3\} + 0.5 \end{bmatrix} = \begin{bmatrix} 2 \\
  3.5 \end{bmatrix} \ldots
  \]
How would you model a transmission link that can transmit packets at a rate $G$ packets per second and has a propagation delay equal to 100ms?