III. Petri Net

Outline

- Petri Net based protocol specification & verification
  - Petri net
  - Behavioral properties
  - Reachability analysis

- Protocol performance analysis

Petri Net Definition

- A directed, weighted, bipartite graph, have 4 basic elements:
  - Places: circles, represents possible "states" of system, may have capacity
  - Tokens: dots, dynamic objects that move between places
  - Transitions: bar or box, events or actions which cause the change of state
  - Arcs: arrows, every arc connects a place with a transition or a transition with a place, may have weight

- A formal definition: A 5-Tuple, \( PN = (P, T, F, W, M_0) \) where
  - \( P = \{p_1, p_2, \ldots, p_m\} \) is a finite set of places,
  - \( T = \{t_1, t_2, \ldots, t_n\} \) is a finite set of transitions,
  - \( F \subseteq (P \times T) \cup (T \times P) \) is a set of arcs, (flow relation)
  - \( W: F \rightarrow \{1, 2, 3, \ldots\} \) is a weight function,
  - \( M_0: P \rightarrow \{0, 1, 2, 3, \ldots\} \) is the initial marking (initial state)
  - \( I(t) = \{p_i \in P : (p_i, t) \in F\} \) Input places
  - \( O(t) = \{p_i \in P : (t, p_i) \in F\} \) Output places
State and State Transition

- The distribution of tokens in a marked Petri Net defines the state of the net, and is called its marking.
  - A marking $M$ is a vector, $M=$(M(P₁), M(P₂), ..., M(Pₙ))
  - $M(P_i)$ is the number of tokens in $P_i$
- The marking may change as a result of firing the transitions.
- In different markings, different transitions may be enabled.

Rules for state transition in Petri Net

- A transition $t$ is said to be enabled if each input place $p$ of $t$ contains at least the number of tokens equal to the weight of the arc from $p$ to $t$ ($w(p, t)$).
- An enabled transition $t$ may or may not fire depending on the additional interpretation (whether or not the event actually takes place).
- A firing of an enabled transition $t$ removes $w(p, t)$ tokens from each input place $p$ of $t$, and adds $w(t, p)$ tokens to each output place $p$ of $t$.

Petri Nets For Modeling

- States of a process are modeled by the distribution of tokens (marking, $M_n$) over the places.
  - Current state: the configuration of tokens over the places
  - Reachable state: a state reachable form the current state by firing a sequence of enabled transitions
  - Dead state: a state where no transition is enabled
- State transitions leading from one state to another are modeled by transitions.
- Input places represent preconditions, input data/signals, resources needed, buffers, channels.
- Transitions represent events, computation, signal processor, task or job, transformations or transportations.
- Output places represent postconditions, output data/signals, resources released, conclusions.

Net Structures

- A sequence of events/actions:
- Concurrent executions:
  - In the real world, events happen at the same time
  - A system may have many local states to form a global state
  - There is a need to model concurrency and synchronization.
Net Structures

- Non-deterministic events - conflict, choice or decision: A choice of either e1, e2 ... or e3, e4 ...

Example: Vending Machine

- The machine dispenses two kinds of snack bars – 20c and 15c
- Only two types of coins can be used: 10c coins and 5c coins
- The machine does NOT return any change (!)

Scenario 1:
- Deposit 5c, 5c, 5c, 5c,
- Take 20c snack bar.

Scenario 2:
- Deposit 10c, 5c, 5c,
- Take 15c snack bar.

Scenario 3:
- Deposit 5c, 10c, 5c,
- Take 20c snack bar.

Example: In a Restaurant

- Scenario 1: Waiter takes order from customer 1; serves customer 1; takes order from customer 2; serves customer 2.
- Scenario 2: Waiter takes order from customer 1; takes order from customer 2; serves customer 2; serves customer 1.
Example: In a Restaurant (Scenario 1)

Customer 1

Waiter

Customer 2

Take order

Order taken

Take order

Wait

Order taken

Serve food

Serve food

eating

eating

Example: In a Restaurant (Scenario 2)

Customer 1

Waiter

Customer 2

Take order

Order taken

Take order

Wait

Order taken

Serve food

Serve food

eating

eating

Example: Reliable Data Transfer Protocol 1

proc. 1

ready to send

send msg.

receive ack.

ack received

buffer full

msg. received

buffer full

send ack.

receive msg.

ready to receive

Example: Reliable Data Transfer Protocol 2

Errors (red)

• Data loss (5, 7)
• Timeout (2, 4)
• Data Retransmission (2-C, 4-E)
• ACK loss (6)
• Timeout (2, 4)
• Data retransmission (2-C, 4-E)
• Reject (8, 9)
• ACK retransmission (8-D, 9-D)
**Behavioral Properties**

- **Reachability** (marking M₀, M₁, ..., Mₙ, ...)
  - Mₙ is reachable from M₀ if there exists a sequence of firings that transform M₀ into Mₙ. Reachability is decidable, but exponential.

- **Boundedness** – prevent from overflows of storage areas
  - A PN is bounded if the number of tokens in each place doesn't exceed a finite number k for any marking reachable from M₀
  - A PN is safe if it is 1-bounded.

- **Liveness** – equivalent to deadlock-free
  - A PN is live if, no matter what marking has been reached, it is possible to fire any transition with an appropriate firing sequence (too strict).
  - Relaxed condition: different levels of liveness defined for a t (L₀=dead, L₁=potentially firable, L₂=k times, L₃=fired infinitely, and L₄=live).

- **Reversibility**
  - A PN is reversible if, for each marking M reachable from M₀, M₀ is reachable from M (too strict).
  - Relaxed condition: a marking M' is a home state if, for each marking M reachable from M₀, M' is reachable from M.

---

**Examples – 1**

- A safe (k=1) but non-live Petri net
- At M₂ only t₄ can be fired, At M₃, no one can be fired
- t₁ & t₂ are L₃-live (infinite), t₃ & t₄ are L₁-live (only once)

---

**Examples – 2**

- An unbounded (p₂) but live Petri net, i.e., t₁ to t₄ are all L₄-live (live)

---

**Examples – 3**

- A safe, non-live Petri net, but is L₁-live
- t₀, t₁, t₂, t₃ are L₀-, L₁-, L₂-, L₃-live, unbounded
Examples – 4

A safe and live Petri net

A unbounded (p4) but live Petri net

Reachability Analysis Based on Coverability Tree

- **Coverability Tree**: tree representation of all possible markings
  - **root** = M₀
  - **nodes** = markings reachable from M₀
  - **arcs** = transition firings, transform one marking to another
- If net is unbounded, then tree is kept finite by introducing the symbol ω
  - For each integer n, ω > n, ω ± n = ω, and ω ≥ ω

Properties

- A PN is **bounded** iff ω doesn’t appear in any node
- A PN is **safe** iff only 0’s and 1’s appear in nodes
- A transition is **dead** iff it doesn’t appear in any arc
- If M is reachable form M₀, then exists a node M' that covers M (M ≤ M')

For a bounded Petri net, the coverability tree (graph) is called the reachability tree (graph)

Algorithm for Coverability Tree Construction

- **Step 1** Label the initial marking M₀ as the root and tag it "new"
- **Step 2** While “new” markings exist, do the following:
  - **Step 2.1** Select a new marking M
  - **Step 2.2** If M is identical to a marking on the path from the root to M, then tag M “old” and goto to another new marking
  - **Step 2.3** If no transitions are enabled at M, tag M “dead-end” (terminal)
  - **Step 2.4** While there exist enabled transitions at M, do the following for each enabled transition t at M:
    - **Step 2.4.1** Obtain the marking M’ that results from firing t at M
    - **Step 2.4.2** On the path from the root to M if there exists a marking M** such that M'(p) = M**(p) for each place p and M' = M**, i.e., M' is coverable, then replace M'(p) with ω
    - **Step 2.4.3** Introduce M' as a node, draw an arc with label t from M to M', and tag M' “new”

Coverability Tree Example –1

Dept. Electrical & Information Engineering, Computer Engineering Lab 3-21 Dr. Junzhao Sun

Dept. Electrical & Information Engineering, Computer Engineering Lab 3-22 Dr. Junzhao Sun

Dept. Electrical & Information Engineering, Computer Engineering Lab 3-23 Dr. Junzhao Sun

Dept. Electrical & Information Engineering, Computer Engineering Lab 3-24 Dr. Junzhao Sun
Coverability Tree Example –1

M₀=(100)
M₁=(001) "dead end"
M₃=(1ω₀)
M₄=(0ω₁)

Dept. Electrical & Information Engineering, Computer Engineering Lab
3-26 Dr. Junzhao Sun
Coverability Tree Example – 1

- PN is unbounded (ω appears @ p2)
- t0 is a dead transition (not live)
- 6 reachable states (M1 is a dead state)

M0=(100)
M1=(001)
M3=(1ω0)
M4=(0ω1)
M6=(1ω0)
M5=(0ω1)

“dead end”
“old”

Coverability Tree Example – 2

- PN is unbounded (ω appears @ p2)
- 6 reachable states (one dead state)

A bounded PN, 7 reachable states, 1 dead state

Coverability Tree Example – 3

coverability graph
coverability tree

A bounded PN, 7 reachable states, 1 dead state
Petri Net – Summary

- PNs have a rich body of knowledge
- PNs are applied successfully to a broad range of problems
  - Concurrent, asynchronous, distributed, parallel, nondeterministic and/or stochastic systems
  - Graphical tool
    - visual communication aid
- PN consists of places, transitions, tokens, arcs
- Behavioral properties include reachability, boundedness, liveness, reversibility, etc.
- Coverability Tree can be used for reachability analysis
- Some extensions: Colored, Timed, Stochastic PN

Outline

- Petri Net based protocol specification & verification
  - Petri net
  - Behavioral properties
  - Reachability analysis
- Protocol performance analysis

Systematic Protocol Performance Analysis

1. State goals, define boundaries
   - Not too general, be concrete, specific, restrictive, measurable, achievable
2. Select performance metrics
   - Related to speed, capability, accuracy, availability of services
3. List system & workload parameters, select factors & values
   - Those affect performance and under study ones, set appropriate levels
4. Select evaluation techniques, workload
   - Technique: depends upon time, resources and desired level of accuracy
   - Workload: set of service requests to system, representative of real life
5. Design experiments, analyze & interpret data, present results
   - Maximize results with minimal effort, balance factors with levels
   - Statistical techniques, discussion and conclusions
   - Result presentation, easy to understand, e.g. graph

Selecting an Evaluation Technique (1/3)

- Three techniques: analytical modeling, simulation, measurement
- What life-cycle stage of the system?
  - Measurement only when something (e.g. prototype) exists
  - If new, analytical modeling or simulation are only options
- When are results needed? (often, yesterday!)
  - Analytic modeling only choice
  - Simulations and measurement can be same
- What tools and skills are available?
  - Skills in analytic modeling (e.g. queuing theory, stochastics)
  - Maybe languages to support simulation
  - Tools to support measurement
    - Example: packet sniffers, source code to add monitoring hooks
Selecting an Evaluation Technique (2/3)

- Level of accuracy desired?
  - Analytic modeling coarse (if it turns out to be accurate, even the analysts are surprised!)
  - Simulation has more details, but may abstract key system details
  - Measurement may sound real, but workload, configuration, etc., may still be missing
    - Accuracy can be high to none without proper design
  - Even with accurate data, still need to draw proper conclusions
    - Ex: so response time is 10.2351 with 90% confidence. So what? What does it mean?

- What are the alternatives? (on problem of selection)
  - Can explore trade-offs easiest with analytic models, simulations moderate, measurement most difficult
    - Ex: QFind – determine impact (tradeoff) of RTT and OS
    - Difficult to measure RTT tradeoff
    - Easy to simulate RTT tradeoff in network, not OS

Selecting an Evaluation Technique (3/3)

- Cost?
  - Measurement generally most expensive
    - Load generator, software/hardware monitor, intrusion to system
  - Analytic modeling cheapest (pencil and paper)
  - Simulation often cheap but some tools expensive
    - Many runs, traffic generators, network simulators

- Salability?
  - Much easier to convince people with measurements
  - Most people are skeptical of analytic modeling results since hard to understand
    - Often validate with simulation before using

- Can use two or more techniques
  - Validate one with another
  - Most high-quality performance analysis papers have analytic model + simulation or measurement

Summary for Evaluation Technique Selection

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Modeling</th>
<th>Simulation</th>
<th>Measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Stage</td>
<td>Any</td>
<td>Any</td>
<td>Prototype+</td>
</tr>
<tr>
<td>2. Time required</td>
<td>Small</td>
<td>Medium</td>
<td>Varies</td>
</tr>
<tr>
<td>3. Tools</td>
<td>Analysts</td>
<td>Some languages</td>
<td>Instrumentation</td>
</tr>
<tr>
<td>4. Accuracy</td>
<td>Low</td>
<td>Moderate</td>
<td>Varies</td>
</tr>
<tr>
<td>5. Trade-off eval.</td>
<td>Easy</td>
<td>Moderate</td>
<td>Difficult</td>
</tr>
<tr>
<td>6. Cost</td>
<td>Small</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>7. Salability</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
</tr>
</tbody>
</table>

Protocol Engineering emphasizes more on Modeling & Simulation.

Selecting Performance Metrics

Metrics: criteria to study performance

- Time
  - Rate
  - Resource

- Speed

- Reliability
  - Availability
  - Probability
  - Time between

- System
  - Done
  - Not Correct
  - Error
  - Event
  - Duration
  - Time between
Types of Workloads

Workloads: requests by users/applications to the system

- **Test workload** – denotes any workload used in performance study
  - Cannot be repeated (easily)
  - May not even exist (proposed system)

- **Real workload** – one observed on a system while being used
  - Cannot be repeated (easily)
  - May not even exist (proposed system)

- **Synthetic workload** – similar characteristics to real workload
  - Can be applied in a repeated manner
  - Relatively easy to port

- **Benchmark** = Workload
  - Standardized or generally accepted workload programs or criteria
  - Benchmarking is process of comparing 2+ systems with workloads

FSM and PN in Performance Analysis

- **FSM extension for performance analysis**
  - DFSMs (deterministic FSM) have at most one trace for any given input string, and N-FSMs may have many alternative traces for a given input
  - Probability FSM (PFMS): there are probabilities labeled for each transition, i.e., a probability that the next state is $s_j$ given that the current state is $s_i$
  - Hidden Markov Models (HMMs), outputs generated by the HMM are generated at states of the FSM, each state emits one of several symbols with some specified probability

- **Petri Nets extensions for performance analysis**
  - Stochastic Petri Net (SPN): a PN that the firing does not occur instantaneously and the firing time associated with each transition is an exponentially distributed random event
  - Generalized Stochastic Petri Net (GSPN): a SPN that both timed transition and instantaneous transitions are allowed
  - Equivalent to continuous time Markov processes

References