Towards a formal verification of the Pastry routing protocol

Tianxiang Lu
Stephan Merz
Christoph Weidenbach

VINO 2011, Campo Tures, Italy

July 15, 2011
Introduction

• **Pastry Routing Protocol**
  - Overlay P2P network protocol
  - Distributed Hash Table
  - Self organized nodes
  - Resilient to churn:
    • concurrent join
    • silent departure

• **Virtual ring**
Introduction

• Verification Challenges
  – Complex data structure
  – Distributed protocol: absence of global state
  – Dynamic network: spontaneous departure, integration of nodes

• In the talk I will show you
  – How we formally modeled Pastry
  – How we debugged and refined the formal model
  – How we prove properties of Pastry
Formal Model in TLA+

vars ≜ (\text{receivedMsgs}, \text{status}, \text{lset}, \text{probing}, \text{failed}, \text{rtable})

Init ≜ \land \text{receivedMsgs} = \{\}
\land \text{status} = [i \in I \mapsto \text{IF } i \in A \text{ THEN } \text{"ready" ELSE } \text{"dead"} ]
\land \text{probing} = [i \in I \mapsto {}]
\land \text{failed} = [i \in I \mapsto {}]
\land \text{lset} = \ldots
\land \text{rtable} = \ldots

Next ≜ \exists i, j \in I : \lor \text{RouteLookup}(i, j) \lor \text{RouteJoin}(i, j) \lor \text{Deliver}(i, j)
\lor \text{Join}(i, j) \lor \text{JReply}(j, i) \lor \text{Probe}(i) \lor \text{PReply}(i, j)
\lor \text{NodeLeft}(i) \lor \text{SuspectFaulty}(i, j) \lor \text{ProbeTimeOut}(i, j)
\lor \text{LSRepair}(i) \lor \text{Recover}(i) \lor \text{ResignNode}(i)
\lor \text{RequestLease}(i) \lor \text{ReceiveLReq}(i) \lor \text{ReceiveBLS}(i)
\lor \text{LeaseExpired}(i, j) \lor \text{DeclareDead}(i, j)

Spec ≜ Init \land \Box[\text{Next}]_{\text{vars}}
Verification Target

• Safety Property: Correct Delivery

  – For each key $k$, there is at most one node $i$ that may deliver, and no ready node is closer to $k$ than $i$.

\[
CorrectDelivery \triangleq \forall i, k \in I:\
\text{ENABLED } Deliver(i, k) \\
\Rightarrow \forall n \in I: \text{status}[n] = \text{"ready"} \Rightarrow \text{AbsDist}(i, k) \leq \text{AbsDist}(n, k) \\
\wedge \forall j \in I \setminus \{i\}: \neg\text{ENABLED } Deliver(j, k)
\]
Model Checking

- Model Checking using TLC, the model checker for TLA⁺
- 2 CPUs
  - 32 Bit Linux machine with Xeon(R) X5460
  - 3.16GHz, 4 GB of memory per CPU
  - Total state space roughly: $2^{152} \times 3^{64}$

<table>
<thead>
<tr>
<th>Property</th>
<th>Time</th>
<th>Depth</th>
<th># states</th>
<th>Counter Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>NeverDeliver</td>
<td>1&quot;</td>
<td>5</td>
<td>101</td>
<td>yes</td>
</tr>
<tr>
<td>NeverJoin</td>
<td>1&quot;</td>
<td>9</td>
<td>19</td>
<td>yes</td>
</tr>
<tr>
<td><strong>CorrectDelivery</strong></td>
<td>5'35&quot;</td>
<td>16</td>
<td>278904</td>
<td><strong>yes</strong></td>
</tr>
</tbody>
</table>
Join

Leaf set range of $i$

$l=2$

Right set

Neighbors of $i$

Left set

Coverage of $i$

Neighbors of $i$

Join($j, s$)

JReply($i, j$)

Probe($j, a_1$)

Probe($j, a_2$)

... Probe($j, a_n$)

PReply($a_1, j$)

PReply($a_2, j$)

... PReply($a_n, j$)

Complete?

no

Repair($j$)

yes

j: “ready”

August 17, 2011
Lease Granting Protocol

[Haebeleren et al. 2005, FreePastry]
Model Checking

- Model Checking using TLC, the model checker for TLA+
- 2 CPUs
  - 32 Bit Linux machine with Xeon(R) X5460
  - 3.16GHz, 4 GB of memory per CPU

<table>
<thead>
<tr>
<th>Property</th>
<th>Time</th>
<th>Depth</th>
<th># states</th>
<th>Counter Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>NeverDeliver</td>
<td>1&quot;</td>
<td>5</td>
<td>101</td>
<td>yes</td>
</tr>
<tr>
<td>NeverJoin</td>
<td>1&quot;</td>
<td>9</td>
<td>19</td>
<td>yes</td>
</tr>
<tr>
<td>CorrectDeliver</td>
<td>&gt; 1 month</td>
<td>21</td>
<td>1952882411</td>
<td>no</td>
</tr>
</tbody>
</table>
Proving Correct Delivery

• To prove: $Spec \Rightarrow []CorrectDelivery$

1. Invent a property $Inv$, in order to apply the rule

$$Spec \Rightarrow []Inv \quad Inv \Rightarrow CorrectDelivery$$

$$\quad Spec \Rightarrow []CorrectDelivery$$

2. Prove $Spec \Rightarrow []Inv$ by:

$$Init \Rightarrow Inv \quad Inv \land A(i, j) \Rightarrow Inv'$$

for every sub-action $A(i, j)$ of $Next$

$$Spec \Rightarrow []Inv$$

• Recall that $Spec = Init \land [][Next]_{vars}$
Current Status of Proof

• Definition of a candidate invariant $\text{Inv}$
  – With the help of TLC and understanding of Pastry

• Proof of $\text{Inv} \Rightarrow \text{CorrectDelivery}$
  – Case analysis on the relative positions of $i$ and $k$
  – In each case proof by contradiction

• Proof of invariant preservation
  – Proof for several (not yet all) sub-actions of Next
TLA⁺ Proof System

- Interactive proof system
- Hierarchical, declarative proof language
- Different automatic back-end provers
- New TLA toolbox IDE released in October 2010 supporting TLAPS
Conclusion + Outlook

• Real-world case study of complex network protocol
  – Modeling routing and join protocols in Pastry
  – Formal Model – Model Checking – Modification (TLA+, TLC)
  – Use of Theorem Proving (TLAPS)

• Next Steps
  – Finish the inductive proof of Invariant
  – Discuss the verification results with algorithm designers

• Outlook
  – Refine the model with more details (e.g. real time, liveness)
  – Improve automation by more powerful backends (arithmetic, first-order logic)
Thank you!

What are you trying to prove?