max planck institut
informatik

Formal verification of Pastry Using TLA+

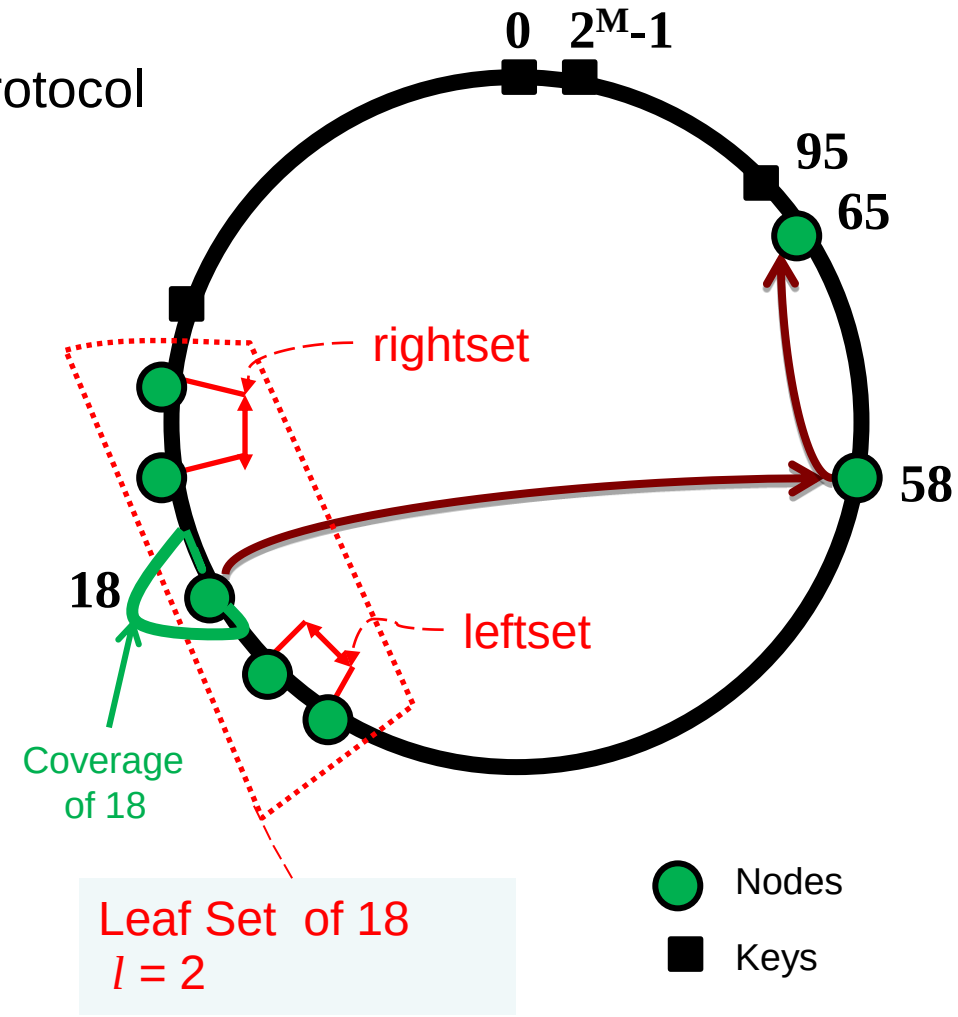
Tianxiang Lu
Stephan Merz
Christoph Weidenbach

TLA+ Workshop at FM2012, Paris

August 27, 2012

Introduction

- Pastry
  - Overlay P2P network protocol
  - Distributed Hash Table
  - Self organized nodes
  - Resilient to churn:
    - concurrent join
    - silent departure
- Virtual ring
  - (see the picture)



Introduction

- Verification Challenges
  - Complex data structure
  - Distributed protocol: absence of global state
  - Dynamic network: spontaneous departure, join of nodes
- Today I will talk about
  - How we formally modeled Pastry in TLA$^+$
  - How we prove properties of Pastry using TLAPS

Formal Model in TLA+

$$
\begin{array}{lcl}
vars & \triangleq & \langle receivedMsgs, status, lset, probing, failed, rtable \rangle \\
Init & \triangleq & \land\ receivedMsgs = \{\} \\
 & & \land\ status = [i \in I \mapsto \text{IF } i \in A \text{ THEN “ready” ELSE “dead”}] \\
 & & \land\ probing = [i \in I \mapsto \{\}] \\
 & & \land\ failed = [i \in I \mapsto \{\}] \\
 & & \land\ lset = \ldots \\
 & & \land\ rtable = \ldots \\
Next & \triangleq & \exists i, j \in I : \lor\ RouteLookup(i,j) \lor RouteJoin(i,j) \lor Deliver(i,j) \\
 & & \quad \lor\ Join(i,j) \lor JReply(j,i) \lor Probe(i) \lor PReply(i,j) \\
 & & \quad \lor\ NodeLeft(i) \lor SuspectFaulty(i,j) \lor ProbeTimeOut(i,j) \\
 & & \quad \lor\ LSRepair(i) \lor Recover(i) \lor ResignNode(i) \\
 & & \quad \lor\ RequestLease(i) \lor ReceiveLReq(i) \lor ReceiveBLS(i) \\
 & & \quad \lor\ LeaseExpired(i,j) \lor DeclareDead(i,j) \\
Spec & \triangleq & Init \land \Box[Next]_{vars}
\end{array}
$$

Verification Target

- Validate model by refuting impossibility claims
  - ***NeverJoin****: A new node can never be joined the network*
  - ***NeverDeliver:*** *A lookup message can never be delivered*

- Safety Property: Correct Delivery
  - *For each key* ***k****, there is at most one node* ***i*** *that may deliver, and no other node is closer to* ***k*** *than* ***i****.*

$$
\begin{aligned}
&\mathit{CorrectDelivery} \triangleq \forall i, k \in I : \\
&\quad \text{ENABLED } \mathit{Deliver}(i,k) \\
&\quad \Rightarrow \ \wedge \forall n \in I : \mathit{status}[n] = \text{“}\mathit{ready}\text{”} \Rightarrow \mathit{AbsDist}(i,k) \leq \mathit{AbsDist}(n,k) \\
&\qquad \wedge \forall j \in I \setminus \{i\} : \neg \text{ENABLED } \mathit{Deliver}(j,k)
\end{aligned}
$$

Model Checking Pastry Properties

- Model Checking using TLC
- Statistics
  - 8 state variables
  - 11 concurrent actions
  - Total state space roughly: 2^{152} X 3^{64} ($\approx 10^{76}$) for 4 nodes
  - Server with 2 CPUs (32 Bit Linux machine with Xeon(R) X5460)
  - 3.16GHz, 4 GB of memory per CPU

Property	Time	Depth	# states	Counter Example
NeverDeliver	1"	5	101	yes
NeverJoin	1"	9	19	yes
……				
CorrectDelivery	**> 1 month**	**21**	**1952882411**	**no**

Proving Correct Delivery

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

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

$$\frac{Spec \Rightarrow []Inv \qquad Inv \Rightarrow CorrectDelivery}{Spec \Rightarrow []CorrectDelivery}$$

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

$$\frac{Init \Rightarrow Inv \qquad Inv \wedge A(i,j) \Rightarrow Inv' \text{ for every sub-action } A(i,j) \text{ of } Next}{Spec \Rightarrow []Inv}$$

- Recall that $Spec = Init \wedge [][Next]_{vars}$

Proof in TLA$^+$ toolbox

- Proof of the model in TLAPS with strong assumptions
  - no nodes leave the network
  - only one node can join the network at a time in any neighboring region
- Statistics
  - 23 invariants proved by induction on 11 actions
  - About 100 lemmas on arithmetic and ring calculation
  - About 100 lemmas on data structures
  - About 1200 proof steps for proving type correctness
  - About 12500 proof steps for inductive proof of invariants
    - CPU Intel Core i3-2330M 2.20GHz, 8 GB RAM, 64-bit, Win7
    - JVM –Xms5120M -Xmx5120M -XX:PermSize=2048M
    - About 10 minutes and 5GB for generating proof obligations

Done & Doing

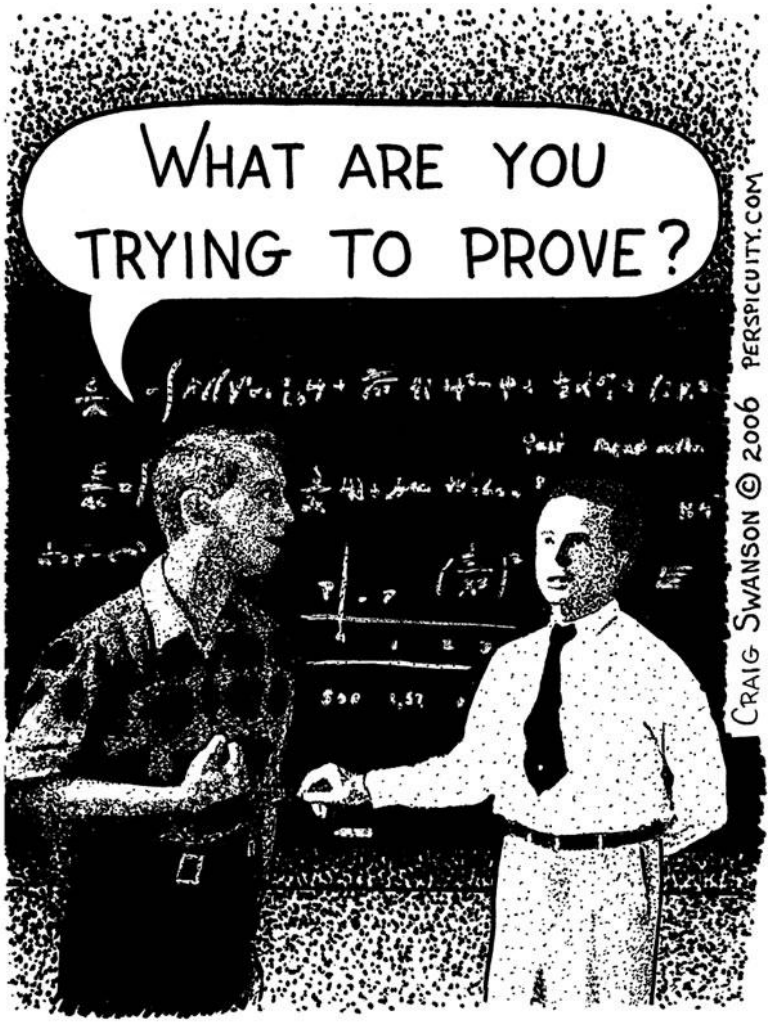
- Done
  - Real-world case study of complex network protocol: Pastry
    - Found bugs in Protocol and improved it.
  - Modeled routing and join protocols in TLA+ and model checked them in TLC
  - Finished the proof of the model in TLAPS with strong assumptions

- Doing
  - Relaxed the assumptions: more nodes join in neighboring region
  - Finding the proper invariants and proving them

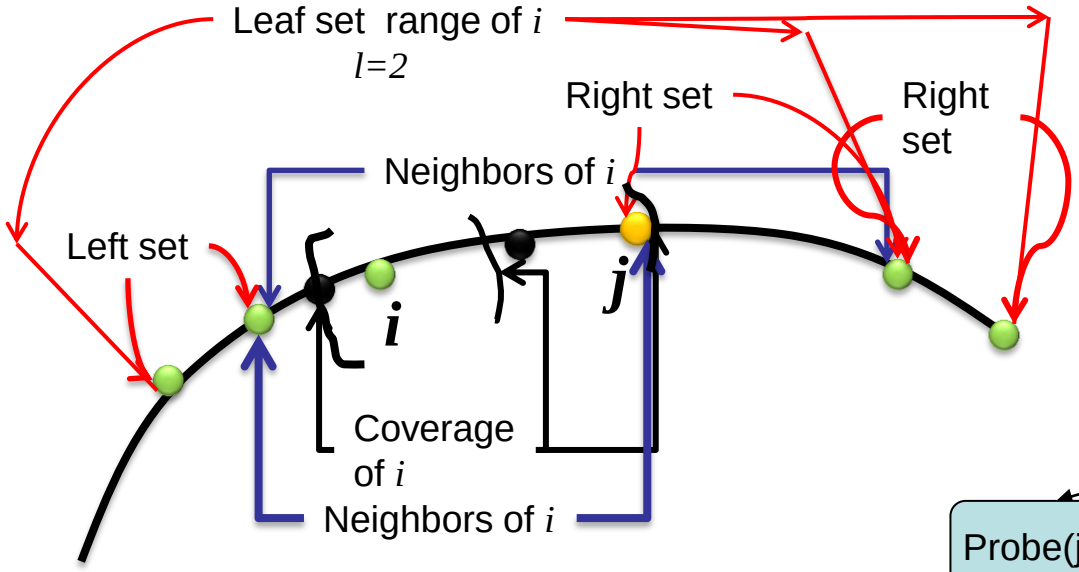
Remarks on the Tools

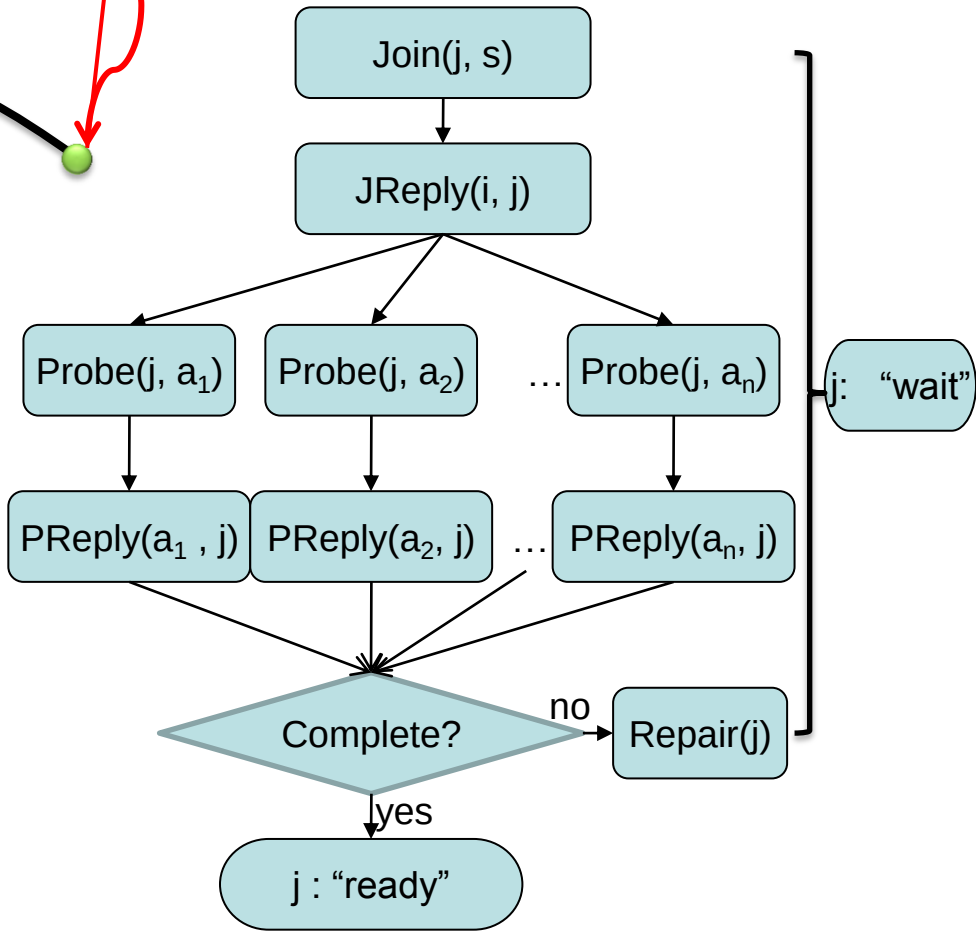
- Trace explorer
  - Very useful !
  - Display the action name ?
- TLC with multi-threads
  - Significant speed up
  - Huge memory footprint and no CPU usage after weeks
    - Java runtime problem ?
    - What about distributed version of TLC ?
- TLAPS
  - Proof editing is very convenient! (zoom, non-linear, jump …)
  - Generation of proof obligation caused memory problem ?

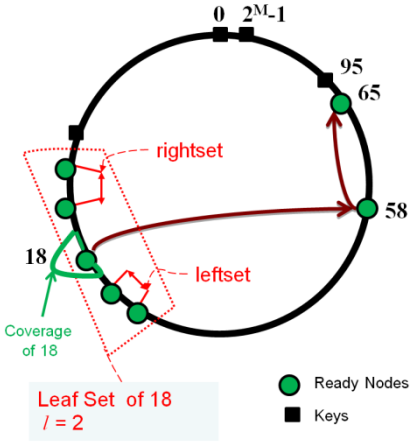
Thank you !


WHAT ARE YOU TRYING TO PROVE?
CRAIG SWANSON © 2006 PERSPICUITY.COM

Join

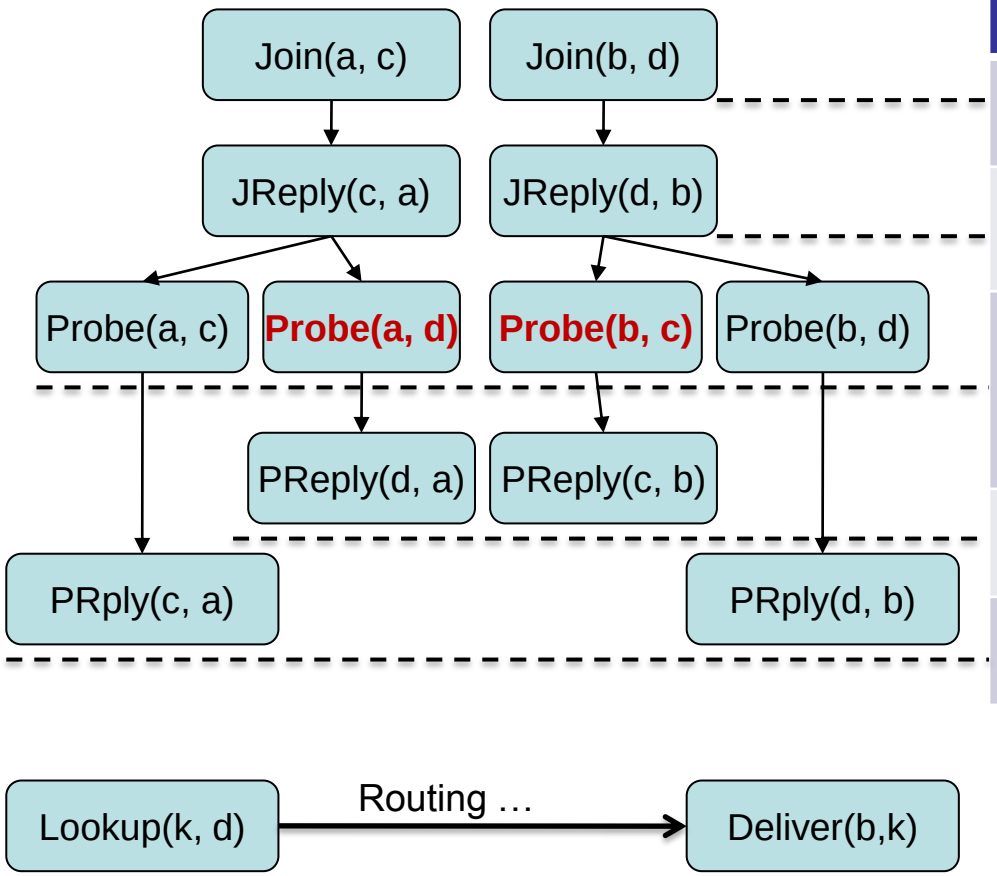




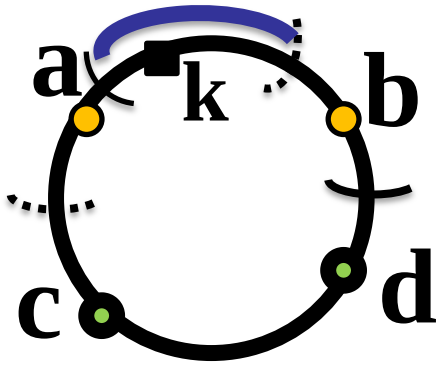


- Waiting node
- Ready node
- Dead node/ Key

Bug of Pastry



ls(a)		ls(b)		ls(c)		ls(d)	
-	-	-	-	d	d	c	c
c	***d***	***c***	***d***	d	d	c	c
c	d	c	d	d	***b***	c	***a***
c	d	c	d	d	b	c	a
c	d	c	d	d	***a***	c	***b***



Lease Granting Protocol

[Haeberlen et al. 2005, FreePastry]