A TLA+ validation of the Chord protocol

Jean-Paul Bodeveix [1] Julien Brunel [2] David Chemouil [2]
Mamoun Filali [1]

IRIT CNRS UPS, Université de Toulouse, France,

ONERA DTIS, Université de Toulouse, France.

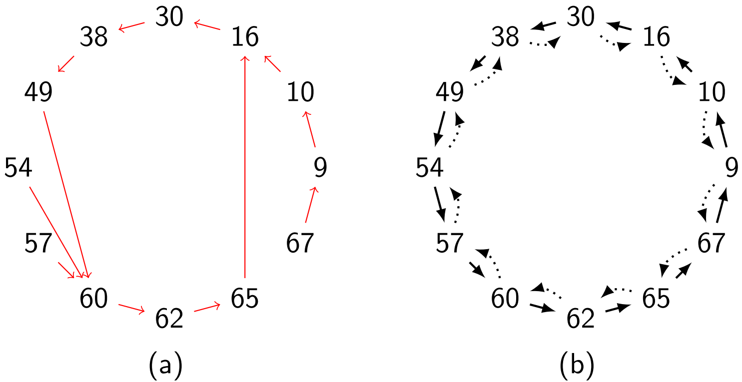
October 2020
TLA+ Community Event

History

- Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications [SMK+01].
- Reasoning About Identifier Spaces: How to Make Chord Correct [Zav17].
- Mechanically Verifying the Fundamental Liveness Property of the Chord Protocol [BBCF19].

We address the Chord maintenance protocol.

The Chord maintenance protocol



Talk

Focus on the verification of a liveness property of the maintenance protocol: stabilization.

- A TLA+ model.
- Validation in the TLA logic.
  - Basic notions and properties.
  - Proof development.
- Mechanization with Isabelle-TLA.

Static description: data structures

(transcription from Isabelle theories)

```
FromPredecessor ≜ 2
pc_chord ≜ {Idle, FromSuccessor, FromPredecessor}
state ≜ [                          \* state of a node
  member : BOOLEAN,                \* is the node alive
  sl : Seq(Nat),                   \* successor list
  prdc : Nat,                      \* predecessor
  inbox : SUBSET Nat,              \* box of delivered messages
  pc : pc_chord,                   \* program counter
  no_more_join_or_fail : BOOLEAN ] \* for stabilization
```

```
State ≜ [Nat → state]     \* global state
```

maintenance protocol ([Zav17]):

- `stabilize`, (protocol action)

  $$stabilize(self) = \mathbf{gc}(stabilize\_guard(self), stabilize\_command(self))$$

- `from_successor`, (protocol action)
- `from_predecessor`, “
- `rectify`, “
- `join`, “
- `fail`, (operating assumptions).
- `no_more_join_or_fail`. (virtual action for stabilization).

$$Spec = \begin{array}{ll} & \exists self \in Nodes : \lor \begin{array}{l} stabilize(self) \\ \ldots \end{array} \\ \land & Liveness \end{array}$$

Liveness

$$
\begin{aligned}
\text{Liveness} \triangleq\ & \\
& \land\ \forall\ n \in \text{Node} : \text{WF\_vars}(\text{stabilize}(n)) \\
& \land\ \forall\ n \in \text{Node} : \text{WF\_vars}(\text{from\_successor}(n)) \\
& \land\ \forall\ n \in \text{Node} : \text{WF\_vars}(\text{from\_predecessor}(n)) \\
& \land\ \forall\ n \in \text{Node} : \forall\ m \in \text{Node}: \text{WF\_vars}(\text{rectify}(n,m))
\end{aligned}
$$

Protocol properties [Zav17]

- Stabilization: *when no more joins of fails occur, all the live nodes : members, are eventually linked through a unique ring. Each node successor list is correct with respect to the member nodes.*
- inductive Invariant: the successor list of *member* nodes of a node is not empty and the set of successor list principal nodes is not empty.

Ring notions

```
between(n1,n2) ≜ \* the set of nodes strictly between n1 and n2
  IF n1 < n2 THEN {nb ∈ Nodes: n1 < nb ∧ nb < n2}
  ELSE {nb ∈ Nodes: n1 < nb ∨ nb < n2}
```

Theorem

Given a non empty set of nodes M, we define the successor function sucNode and the predecessor function prevNode.

```
sucNode[M ∈ SUBSET Nat, n ∈ Nat] ≜
  (IF M = {n} THEN n
   ELSE IF {k ∈ M: k > n} = ∅ THEN Min({k ∈ M: k < n})
   ELSE Min({k ∈ M: k > n}))
```

Principals

Definition

Given a set of nodes M, a function f over M, the principals of f are the nodes of M that are not between by any pair $(m, f(m))$.

principals (M,f) $\triangleq$ {p $\in$ M: $\forall$ m $\in$ M: p $\notin$ between(m, f[m])}

NB. These principals are not *sucessor lists principals.* These principals are defined over functions from M to M. We introduce them to decompose the proof of stabilization.

$$sl\_principals(sl \circ St) \subseteq principals(First(St))$$

Theorem (all_principals)

Given a function f over the set of nodes M, M is the set of principals iff f is the sucNode function over M.

```
THEOREM all_principals ≜
  ASSUME NEW M, NEW f,
         M ⊆ Nodes, ∀ e ∈ M: f[e] ∈ M
  PROVE       (M = principals(M,f))
                       ⇔
        (∀ m ∈ M: f[m] = sucNode[M, m])
```

Theorem (prevNode_is_principal)

Given a function f over the set of nodes M, p a principal of f, the prevNode of p over M is also a principal of f iff the only node in M with image p is the prevNode of p over M.

THEOREM *prevNode_is_principal* $\triangleq$
  ASSUME NEW *M*, **NEW** *f*, **NEW** *p*,
    $M \subseteq Nodes, \forall\, e \in M: f[e] \in M,\ p \in principals(M,f)$
  PROVE $(\forall\, q \in M: f[q] = p \Leftrightarrow q = prevNode[M,p])$
$$\Leftrightarrow$$
    $(prevNode[M,p] \in principals(M,f))$

Definition (Back propagation of a predicate.)

Given a node *p*, and an indexed state predicate P, we define the back propagation of P, from *p*, over *cnt* hops as the conjunction of the back *cnt* instantiations of *P* starting from *p*.

Definition (Back propagation of a predicate.)

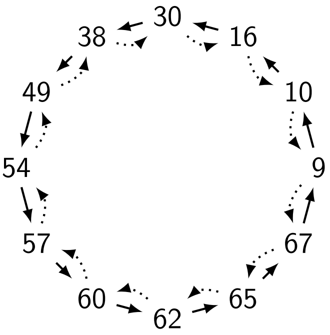
Given a node *p*, and an indexed state predicate P, we define the back propagation of P, from *p*, over *cnt* hops as the conjunction of the back *cnt* instantiations of *P* starting from *p*.

```
propagate_back_over_ring  (M,P,cnt,p) ≜
    \* M member nodes
    \* P   : indexed state predicate to propagate
    \* cnt : number of back propagations
    \* p   : propagation starting point
    [St ∈ State ↦ ∀ j:  j ≤ cnt ⇒ P[prevNode[M]^j[p],St]]
```

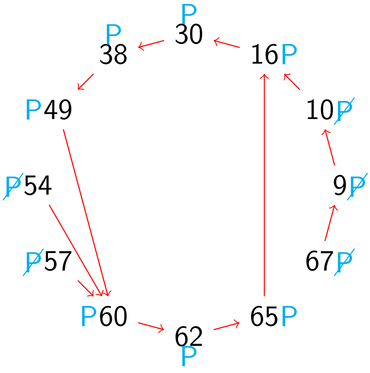
Theorem (Full propagation of a predicate.)

Given a node p, and an indexed state predicate P, the back propagation of P, from p, over $Cardinality(M) - 1$ hops defines actually the full propagation of P over M.

THEOREM *propagate_full* $\triangleq$
  ASSUME NEW *M*, **NEW** *p*, **NEW** *P*,
    $M \subseteq Nodes,\ p \in M$
  PROVE
    $propagate\_back\_over\_ring(M, P, Cardinality(M) - 1, p, St) = (\forall\ q \in M : P[q, St])$



(c)



(d)

- N = 100 Nodes = 0..99
- ⟶ sucNode
- ⇠ prevNode
- example: between (10,16) = 11..15

What do we verify ?

When no more fails or joins occur, eventually:

- a *distributed and replicated* version of the `sucNode` function is built. On each node *n*:
  - the first element of the successor list defines $sucNode[members(St), n]$.
  - the tail of the list defines replicated first successors:
- a *distributed* version of the `prevNode` function is built. On each node *n*: the variable *prdc* defines $prevNode[members(St), n]$.

```
Correctness(St) ≜
   ∧ ∀ p ∈ members(St): First(St,p) = sucNode[members(St),p] \* distribution
   ∧ ∀ p ∈ members(St): ∀ j ∈ 2..L:                          \* replication
         St[p].sl[j] = sucNode[members(St),St[p].sl[j−1]]
   ∧ ∀ p ∈ members(St): St[p].prdc = prevNode[members(St),p] \* distribution
```

Stabilization proof

System invariants [Zav17]:

- the successor list of *member* nodes of a node is not empty.
- the set of successor list principal nodes is not empty.

Stabilization proof phases:

no more joins or fails virtual action.

$\rightsquigarrow$ First elements of successor lists are members

- $\rightsquigarrow$ prevnode delivered to principal
- $\rightsquigarrow$ prdc updates to prevnode
- $\rightsquigarrow$ prevnode becomes principal
  - $\rightsquigarrow$ all members become principal $\rightsquigarrow$ stabilization

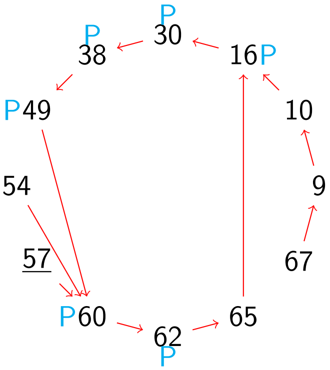


Figure: prevnode (57) delivered to principal (60)

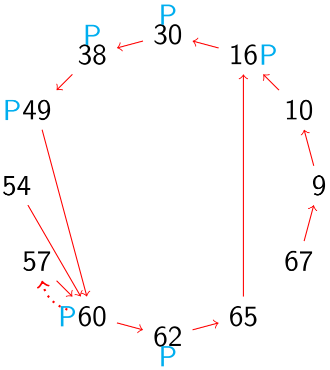


Figure: prdc of 60 updates to prevnode (57)

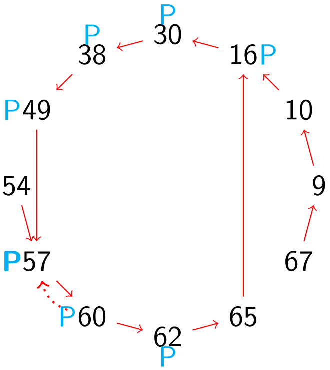


Figure: prevnode (57) becomes principal

Isabelle-TLA

The model and the proofs have been done with Isabelle-TLA.

- State predicates had to be made explicite for better proof automation.
- Transition structuring as *guarded commands* made easier the handling of `Enabled`.
- Ad hoc versions of Meta theorems for liveness thanks to Isabelle-TLA.

Ad hoc metatheorem

$$\frac{\begin{array}{c}\textbf{stable}(\mathit{Next}, \mathit{Phase}) \\ \vdash \textbf{wp}(\mathit{Phase} \wedge P \lhd \mathit{Next}, P \vee Q) \\ \mathit{Phase} \wedge P \wedge \mathit{from\_pred\_G}(\mathit{self}) \wedge \textbf{changes}(\mathit{from\_pred\_C}(\mathit{self})) \\ \rightarrow (Q \circ (\mathit{from\_pred\_C}(\mathit{self}))) \\ \mathit{Phase} \wedge P \rightarrow \mathit{from\_pred\_G}(\mathit{self})\end{array}}{\vdash \mathit{Spec} \rightarrow \mathit{Phase} \wedge P \leadsto Q}$$

- Instantiation of the TLA logic WF rule.
- relies on the fairness of the *from_pred* transition.

Conclusion

- Principals theory (in Isabelle-HOL).
- Isabelle-TLA for temporal properties and Meta theorems.
- Study of the maintenance of the Chord protocol.
  - TLA+ model.
  - [Zav17] invariant is sufficient for stabilization verification.
  - Stabilization liveness relies on the weak fairness of node transitions.

Jean-Paul Bodeveix, Julien Brunel, David Chemouil, and Mamoun Filali.
Mechanically Verifying the Fundamental Liveness Property of the Chord Protocol.
In *23rd Int. Symp. on Formal Methods*, Portugal, October 2019.

Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan.
Chord: A scalable peer-to-peer lookup service for internet applications.
SIGCOMM Comp. Com. Rev., 31(4):149–160, August 2001.

Pamela Zave.
Reasoning about identifier spaces: How to make Chord correct.
IEEE Transactions on Software Engineering, 43(12):1144–1156, Dec 2017.