Propagation of Information with Feedback on Trees: The Proof of Ideal Stabilization

Mikhail Nesterenko[1] and Sébastien Tixeuil[2]

[1] Kent State University
[2] Université Pierre et Marie Curie - Paris 6

Technical report TR-KSU-CS-2012-01, Kent State University

Abstract. We describe an program that implements propagation of information with feedback on rooted trees and prove it ideally and self-stabilizing.

1 Algorithm Description

We describe a program $\mathcal{PIF}$ that ideally stabilizes to the propagation of information with feedback [1, 3] specification. This description is meant as a companion for the PIF presentation in the paper that introduced ideal stabilization [2]. Therein, the program was presented for chain topologies only. This paper expands the presentation and proof for arbitrary rooted trees. Refer to the original paper [2] for motivation, terminology and notation used in this paper.

The program is designed for rooted trees. A *root* is an arbitrary distinguished process in the tree. A non-root process with a single neighbor is *leaf*. Processes that are neither roots nor leaves are *intermediate*. For each process u, all processes that lie on the path from u to the root are *ancestors* of u. Process v is a *descendant* of process u, if u is an ancestor of v. Observe that the root is the ancestor of the all other nodes while all of them are the root's descendants. A *parent* of a process is its nearest ancestor. The *height* of a process is the distance to this node's farthest descendant. A *child* of a process is its nearest descendant. A process may have only one parent but many children. A *causality chain* for a leaf is the path of its ancestors to the root. For any two leaves the causality chains have at least one process, the root, in common. Once we are reasoning about a particular causality chain, we assume that it is laid out horizontally with the root located on the left and the leaf on the right.

The tree is organized as follows. Each process has unique identifier throughout the system. For each process, one neighbor's identifier is designated as its parent. This topological designation is constant and incorruptible. If there is no parent, the process is the root. As a shorthand, we assume that the root just has identifier *root*; a leaf's identifier is $leaf$, each process u has set of neighbors $Ch.u$ that are its children and a constant *parent*.

Each process has a state variable *st*. In the intermediate processes, the variable may hold one of the three values: **i**, **rq**, **rp** which stand respectively for *idle*, *requesting* and *replying*. The root can only be idle or requesting while a leaf can be either idle or replying.

The objective of the program is to send a signal from the root to the leaves and, in return, receive an acknowledgment that matches this signal. Operationally, the program should ensure that after the root makes a request then intermediate processes propagate this request in causally ordered steps along each causality chain transitioning from idle to requesting. Afterwards, the intermediate processes propagate the replys from each leaf back to the root in casually ordered steps transitioning from requesting to replying.

We define the following state predicates. For each causality chain whose length is N, $RP(k)$ are the specification states where all k processes on the left are requesting ($k = 1, N-1$) and the rest of the processes are replying. $RQ(l, m)$ are the specification states where all l processes on the left are requesting ($l = 0, N-1$) and all $m-l$ processes following them are idle ($m = l+1, N$), while the remaining processes are replying. The two predicates are mutually exclusive. Specification $\mathcal{SPIF}$ includes the sequences where, for each causality chain, the system satisfies one of the predicates and transitions from one to the other infinitely.

Observe that a step of the root moves the chain from RP to RQ while a step of the leaf moves the chain back from RQ to RP. Since the root is present in each causality chain, the transition from RP to RQ happens in all chains simultaneously, the transition back to RP may differ for each individual chain.

Let us define another pair of predicates. Predicate $RP'(k)$ defines the states where k processes on the left are requesting ($k = 1, N-1$), the process $k+1$ is replying and the state of the other processes is arbitrary. Notice that $RP(k) \subset RP'(k)$. Predicate $RQ'(l, m)$ are the states where all l processes on the left are requesting ($l = 0, N-1$), all $m-1$ following them are idle ($m = l+1, N$) and the state of the other processes is arbitrary. Similarly, $RQ(l, m) \subset RQ'(l, m)$. Specification $\mathcal{IPIF}$ includes the sequences where the system always satisfies either $RP'(k)$ or $RQ'(l, m)$, each sequence has a sequence in $\mathcal{SPIF}$ as a suffix and the transition from $RQ'(l, m)$ is only to $RP(k)$.

We now describe program $\mathcal{PIF}$. It has only external variables. The mapping between the program and specification states is identical. The actions of $\mathcal{PIF}$ are shown in Figure 1.

2 Correctness Proof

Lemma 1. For any causality chain in the tree, predicate $RQ(l, m) \vee RP(k)$ is closed in $\mathcal{PIF}$.

Proof: (Sketch) Notice that the actions of the processes outside the particular causality chain do not have their variables in the predicate and, therefore, cannot affect it. The closure can then be ascertained by examining the actions of processes in the causality chain. If the program state conforms to $RQ(l, m)$, then the execution of any of the enabled action of a process in this chain moves the system to a state that conforms to either $RQ(l, m)$ or to $RP(k)$. $\square$

$$\begin{array}{llll}
\textit{request}: & st.root = \mathbf{i} & \wedge\ (\forall q \in Ch.root : st.q = \mathbf{i}) & \longrightarrow st.root := \mathbf{rq} \\
\textit{clear}: & st.root = \mathbf{rq} & \wedge\ (\forall q \in Ch.root : st.q = \mathbf{rp}) & \longrightarrow st.root := \mathbf{i} \\
\textit{forward}: & st.parent = \mathbf{rq} \wedge st.p = \mathbf{i} & \wedge\ (\forall q \in Ch.p : st.q = \mathbf{i}) & \longrightarrow st.p := \mathbf{rq} \\
\textit{back}: & st.parent = \mathbf{rq} \wedge st.p = \mathbf{rq} & \wedge\ (\forall q \in Ch.p : st.q = \mathbf{rp}) & \longrightarrow st.p := \mathbf{rp} \\
\textit{stop}: & st.parent = \mathbf{i} \quad \wedge st.p \neq \mathbf{i} & & \longrightarrow st.p := \mathbf{i} \\
\textit{reflect}: & st.parent = \mathbf{rq} \wedge st.leaf = \mathbf{i} & & \longrightarrow st.leaf := \mathbf{rp} \\
\textit{reset}: & st.parent = \mathbf{i} \quad \wedge st.leaf = \mathbf{rp} & & \longrightarrow st.leaf := \mathbf{i}
\end{array}$$

Fig. 1. $\mathcal{PIF}$ program actions. Actions *request* and *clear* belong to the root process; actions *forward*, *back*, and *stop* – to an intermediate processes; actions *reflect* and *reset* – to a leaf.

Lemma 2. For any causality chain, if a computation of $\mathcal{PIF}$ starts in a state conforming to $RQ(l, m)$, this computation also contains a state satisfying $RP(k)$.

Proof: Suppose that initially the program state satisfies $RQ(l, m)$. We show that if $l < N - 1$, eventually both l and m are incremented. If $m < N$, *stop* is enabled at process $m + 1$ in the causality chain. The execution of this action increments m in $RQ(l, m)$.

The case of l is a bit more involved. If $l = 0$, the root is idle. The children of the root may or may not be idle. However, if there is a process $q \in Ch.root$ such that $st.q \neq \mathbf{i}$, then *stop* is enabled in q. The execution of this action transitions q to idle. Once all of the root's children are idle, its *request* action is enabled. If it is executed, root transitions to requesting state which increments l. Let us examine the case of $0 < l < N - 1$. If $PQ(l, m)$ is satisfied, process p_{l+1} and its parent are idle. Similar to the case of the root, if p_{l+1} is idle, *stop* is enabled in its each child that is not idle. Once, every child executes *stop*, *forward* becomes enabled in p_{l+1}. If this action is executed, p_{l+1} becomes requesting and l is incremented. That is, if a state of the causality chain satisfies $RQ(l, m)$ both l and m are eventually incremented.

If $l = N - 1$ and $RQ(l, m)$ is satisfied, all processes in the causality chain but the leaf are requesting while the leaf is idle. In this case *reflect* is enabled in the leaf. The execution of this action moves the leaf to the replying state. In this case, the causality chain satisfies $RP(k)$ with $k = N - 1$. That is, the computation that starts in a state that satisfies $RQ(l, m)$ also contains the state satisfying $RP(k)$ $\square$

Lemma 3. For any causality chain, if a computation of $\mathcal{PIF}$ starts in a state conforming to $RQ'(l, m)$, this computation also contains a state satisfying $RP(k)$.

The proof of this lemma is similar to the proof of Lemma 2.

Lemma 4. For any causality chain in the tree, if a computation of $\mathcal{PIF}$ starts in a state conforming to $RP(k)$, it contains a state conforming to $RQ(l, m)$.

Proof: We first demonstrate that if the computation starts in a state where the causality chain satisfies $RP(k)$ with $k > 1$, then it also contains a state where

k is decremented. We do it by strong induction on the height of all causality chains in the tree. Specifically, we show that every causality chain of length N reaches a state where the process whose distance from the leaf is $N - k + 1$ is replying. The base case of the distance being zero, i.e. $k = N - 1$, means that only leaf is replying. It follows from Lemma 2.

Let us assume that every process at height at least $N - k$ is replying. Let us consider a process p that is not replying and whose height $N - k + 1$. The height of all children of p is at most $N - k$. Due to the assumption, there is a state in this computation where every child of p is replying. Since the chain conforms to $RP(k)$, both p and *parent.p* are requesting. replying. In this case, *back* action is enabled in p. Once executed, p is replying. By induction this proves our claim.

From this claim, it follows that this computation contains a state where all children of the root process are replying. Since this state conforms to $RQ(k)$, The root is requesting. In this case, action *clear* is enabled in the root process. Once, executed, all causality chains of the system transition to $RQ(0, 1)$. Hence, the lemma. □

Lemma 5. For any causality chain in the tree, if a computation of $\mathcal{PIF}$ starts in a state conforming to $RP'(k)$, it contains a state conforming to $RQ(l, m)$

The proof of this lemma is similar to the proof of Lemma 4

Theorem 1. $\mathcal{PIF}$ classically stabilizes to $\mathcal{SPIF}$ and ideally stabilizes to $\mathcal{IPIF}$.

Proof: Specification $\mathcal{SPIF}$ requires that for every causality chain in the tree, the solution should remain in the disjunction of the predicates $RQ(l, m)$ and $RP(k)$ and infinitely transition from one to the other. According to Lemma 1, the disjunction of predicates is closed in $\mathcal{PIF}$. According to Lemmas 2 and 4, if the computation starts in a state conforming to one of the predicates, it transitions to the other. Hence, the disjunction of the predicates is the invariant of $\mathcal{PIF}$ with respect to $\mathcal{SPIF}$.

Observe that for every causality chain, the disjunction of predicates $RQ'(l, m) \vee RP'(k)$. That is, it contains the program and specification state space. According to Lemma 3, if a computation of $\mathcal{PIF}$ starts in a state conforming to $RQ'(l, m)$, then it also contains a state satisfying $RP(k)$. Similarly, due to Lemma 5, if $\mathcal{PIF}$ starts from a state conforming to $RP'(k)$, it transitions to $RQ(l, m)$.This means that the program stabilizes to $\mathcal{SPIF}$ and ideally stabilizes to $\mathcal{IPIF}$. □

References

1. Ernest J. H. Chang. Echo algorithms: Depth parallel operations on general graphs. *IEEE Transactions on Software Engineering*, 8(4):391–401, July 1982.
2. Mikhail Nesterenko and Sébastien Tixeuil. Ideal stabilization. In *AINA*, pages 224–231, March 2011.
3. A. Segall. Distributed network protocols. *IEEE Transactions on Information Theory*, IT-29(1):23–35, January 1983.