Synthesis of Distributed Algorithms with Parameterized Threshold Guards

Igor Konnov

Marijana Lazić    Josef Widder    Roderick Bloem

OPODIS’17, December 2017
Verifying fault-tolerant systems

**safety critical systems**: cars, planes, etc.

- rare but dangerous faults
- 3 to 7 processes

finite-state model checking

**datacenters**: thousands of computers

- faults happen every day
- 100–10,000 processes

parameterized model checking
Fault-tolerant distributed algorithms

\( n \) processes communicate by sending messages

\( f \) processes are faulty (unknown)

\( t \) is an upper bound on \( f \) (known)

resilience condition on \( n, t, \) and \( f \), e.g., \( n > 3t \land t \geq f \geq 0 \)
Reliable broadcast service (informally)

one process broadcasts a message \texttt{bcast}

**correctness**: if all correct processes received \texttt{bcast}, then some correct process \textit{eventually accepts} \texttt{bcast}

**relay**: if a correct process accepts \texttt{bcast}, then all correct processes \textit{eventually accept} \texttt{bcast}

**unforgeability**: if no correct process received \texttt{bcast}, then no correct process ever \textit{accepts} \texttt{bcast}

**fairness**: every sent message is eventually received
local $myval_i \in \{0, 1\}$ - did process $i$ receive bcast?

while true do
  if $myval_i = 1$ and not sent ECHO before
  then send ECHO to all

  if received ECHO from at least $t + 1$ distinct processes
  and not sent ECHO before
  then send ECHO to all

  if received ECHO from at least $n - t$ distinct processes
  then accept

od

resilience: of $n > 3t$ processes, $f \leq t$ processes are Byzantine
local \( myval_i \in \{0, 1\} \) - did process \( i \) receive bcast?

while true do

if \( myval_i = 1 \) and not sent ECHO before
then send ECHO to all

if received ECHO from at least \( t + 1 \) distinct processes
and not sent ECHO before
then send ECHO to all

if received ECHO from at least \( n - t \) processes
then accept

od

resilience: of \( n > 3t \) processes, \( f \leq t \) processes are Byzantine
More threshold guards...

<table>
<thead>
<tr>
<th>Protocol Type</th>
<th>Formula</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliable broadcast</td>
<td>$x \geq t + 1$  \hspace{1cm} $x \geq n - t$</td>
<td>[Srikanth, Toueg’86]</td>
</tr>
<tr>
<td>Hybrid broadcast</td>
<td>$x \geq t_b + 1$  \hspace{1cm} $x \geq n - t_b - t_c$</td>
<td>[Widder, Schmid’07]</td>
</tr>
<tr>
<td>Byzantine agreement</td>
<td>$x \geq \lceil \frac{n}{2} \rceil + 1$</td>
<td>[Bracha, Toueg’85]</td>
</tr>
<tr>
<td>Non-blocking atomic commitment</td>
<td>$x \geq n$</td>
<td>[Raynal’97], [Guerraoui’01]</td>
</tr>
<tr>
<td>Condition-based consensus</td>
<td>$x \geq n - t$  \hspace{1cm} $x \geq \lceil \frac{n}{2} \rceil + 1$</td>
<td>[Mostéfaoui, Mourgaya, Parvedy, Raynal’03]</td>
</tr>
<tr>
<td>Consensus in one communication step</td>
<td>$x \geq n - t$  \hspace{1cm} $x \geq n - 2t$</td>
<td>[Brasileiro, Greve, Mostéfaoui, Raynal’03]</td>
</tr>
<tr>
<td>Byzantine one-step consensus</td>
<td>$x \geq \lceil \frac{n+3t}{2} \rceil + 1$</td>
<td>[Song, van Renesse’08]</td>
</tr>
</tbody>
</table>

In general, there is a resilience condition, e.g., $n > 3t, n > 7t$
Byzantine model checker

[source code, benchmarks, virtual machines, etc.]

10 parameterized fault-tolerant distributed algorithms:

[CAV’15] & [POPL’17]
From verification to synthesis
Different threshold guards for one sketch

local \( myval_i \in \{0, 1\} \) - did process \( i \) receive bcast?

while true do
  if \( myval_i = 1 \) and not sent ECHO before
     then send ECHO to all
  if received ECHO from at least \( t + 1 \) distinct processes
     and not sent ECHO before
     then send ECHO to all
  if received ECHO from at least \( n - t \) distinct processes
     then accept
  od

resilience: of \( n > 3t \) processes, \( f \leq t \) processes are Byzantine
Different threshold guards for one sketch

```plaintext
local \( myval_i \in \{0, 1\} \) - did process \( i \) receive \texttt{bcast}?

while true do
    if \( myval_i = 1 \) and not sent ECHO before
    then send ECHO to all

    if received ECHO from at least \( t + 1 \) distinct processes
    and not sent ECHO before
    then send ECHO to all

    if received ECHO from at least \( n - t \) distinct processes
    then accept

od
```

resilience: of \( n > 3t \) processes, \( f \leq t \) processes are Byzantine

Igor Konnov 9 of 32
Different threshold guards for one sketch

\text{local} \quad myval_i \in \{0, 1\} \quad - \text{did process } i \text{ receive } \text{bcast}?

\text{while true do}
  \quad \text{if } myval_i = 1 \text{ and not sent ECHO before}
  \quad \quad \text{then send ECHO to all}

  \quad \text{if received ECHO from at least } t + 1 \text{ distinct processes}
  \quad \quad \text{and not sent ECHO before}
  \quad \quad \text{then send ECHO to all}

  \quad \text{if received ECHO from at least } n - t \text{ distinct processes}
  \quad \quad \text{then accept}
\text{od}

\text{resilience: of } n > 3t \text{ processes, } f \leq t \text{ processes are Byzantine}
Different threshold guards for one sketch

local $myval_i \in \{0, 1\}$ - did process $i$ receive bcast?

while true do
  if $myval_i = 1$ and not sent ECHO before
    then send ECHO to all

  if received ECHO from at least $t + 1$ distinct processes
    and not sent ECHO before
    then send ECHO to all

  if received ECHO from at least $n - t$ distinct processes
    then accept

od

resilience: of $n > 3t$ processes, $f \leq t$ processes are Byzantine
Find thresholds automatically?

local \( myval_i \in \{0, 1\} \) - did process \( i \) receive \texttt{bcast}?

while true do
  if \( myval_i = 1 \) and not sent ECHO before
  then send ECHO to all

  if received ECHO from at least \(?_1 \cdot n + ?_2 \cdot t + ?_3\) distinct processes
    and not sent ECHO before
  then send ECHO to all

  if received ECHO from at least \(?_4 \cdot n + ?_5 \cdot t + ?_6\) distinct processes
  then accept
od

resilience: of \( n > 3t \) processes, \( f \leq t \) processes are Byzantine
Synthesis loop

Find $?_1, \ldots, ?_k \in \mathbb{Q}$

Generator
infinite
search space

counterexample
coefficients

Verifier
Byzantine MC
	solution
Synthesis problem

Find a distributed algorithm that satisfies spec $\varphi$
for all parameter values $n$, $t$, and $f$ that satisfy resilience condition

Input:

- sketch algorithm,
- specification, e.g., unforgeability, correctness & relay
- resilience condition, e.g., $n > 3t$, $t \geq f \geq 0$

Find (if exist): $a_1, \ldots, a_k \in \mathbb{Q}$ for $\varphi$
Formalizing pseudo-code

Sketch threshold automata to capture the pseudo-code

Linear temporal logic to formalize the specifications

Linear integer arithmetic to express the resilience condition
Sketch threshold automata

Byzantine faults:
run $n - f_b$ processes,
count messages modulo Byzantine processes, e.g., $x + f_b \geq \tau_2$

\[
\begin{align*}
\tau_1 &= \tau_2 \cdot n + \tau_3 \\
\tau_2 &= \tau_4 \cdot n + \tau_5 \cdot t_b + \tau_6
\end{align*}
\]
Sketch threshold automata

\[ f_b \leq t_b \textbf{ Byzantine and } f_c \leq t_c \textbf{ crash faults:} \]

- run \( n - f_b \) processes,
- resilience condition e.g. \( n > 3t_b + 2t_c \land t_b \geq 0 \land t_c \geq 0 \)
Linear temporal logic

**Relay:**
if a correct process accepts \texttt{bcast},
then all correct processes eventually accept \texttt{bcast}
Linear temporal logic

 Relay:
- if a correct process accepts $bcast$, then all correct processes eventually accept $bcast$ and at least one process never accept $bcast$
Linear temporal logic

¬Relay:

if a correct process accepts \textit{bcast}
then all correct processes eventually accept \textit{bcast}
and at least one process never accepts \textit{bcast}

\[
\begin{align*}
E(F(\kappa_{\text{ACCEPT}} \neq 0) \land G((\kappa_{V1} \neq 0 \lor \kappa_{V0} \neq 0 \lor \kappa_{\text{SENT}} \neq 0)) \land GF \psi_{\text{fair}})
\end{align*}
\]
Linear temporal logic

¬Relay:
if a correct process accepts \texttt{bcast}
then all correct processes eventually accept \texttt{bcast}
and at least one process never accepts \texttt{bcast}

\[
E \left( F(\kappa_{\text{ACCEPT}} \neq 0) \land G(\kappa_{\text{V1}} \neq 0 \lor \kappa_{\text{V0}} \neq 0 \lor \kappa_{\text{SENT}} \neq 0) \right) \land G F \psi_{\text{fair}}
\]

Propositional formulas:

(1) \[ \land_{\ell \in S} \kappa_{\ell} = 0 \]
(2) \[ \lor_{\ell \in S} \kappa_{\ell} \neq 0 \]
(3) \[ \land_{S \subseteq T} \lor_{\ell \in S} \kappa_{\ell} \neq 0 \]
(4) \[ \text{Bool} (\text{Guards}) \rightarrow (1) \land (2) \land (3) \]

Temporal formulas:

\[ \psi ::= \text{prop} \mid G \psi \mid F \psi \mid \psi \land \psi \]

We call this fragment \text{ELTL}_{\text{FT}}
Our solution to synthesis
Synthesis loop

Find $\varphi_1, \ldots, \varphi_k \in \mathbb{Q}$

Generator
infinite search space

Verifier
Byzantine MC
coefficients
counterexample
solution

Termination?
sane guards $\Rightarrow$ bounded search space

Efficiency?
Generator learns from counterexamples
Sane guards: thresholds lie in the interval $[0, n]$

Classic threshold guards:

- if received $\frac{n}{2}$ messages... ✓ wait for a majority
- if received $t + 1$ messages... ✓ wait for a correct process
- if received $n - t$ messages... ✓ wait for non-faulty processes

Syntactically correct but meaningless guards:

- if received $2n$ messages... ✗
- if received $-5$ messages... ✗
Search space for sane guards

Resilience condition: \( n > 3t > 0 \)

Threshold: \( 0 \leq ?an + ?bt + ?c \leq n \)

\[
\begin{align*}
0 & \leq ?a & \leq 1 \\
-4 & \leq ?b & \leq 4 \\
-8 & \leq ?c & \leq 8 \\
\end{align*}
\]

Theorem

Assume: \( n > \sum_{1 \leq i \leq k} \delta_i \cdot t_i \) and \( \forall i. t_i \geq 0 \) — resilience cond.

\[
0 \leq ?an + \sum_{1 \leq i \leq k} ?b_i \cdot t_i + ?c \leq n \\
\] — threshold

Then:

\[
\left\{
\begin{array}{c}
0 \leq ?a \leq 1 \\
-B_i \leq ?b_i \leq B_i \quad \text{for } B_i = \delta_i + 1 \quad \text{and} \quad 1 \leq i \leq k \\
-C \leq ?c \leq C \quad \text{for } C = k + 1 + 2(\delta_1 + \cdots + \delta_k)
\end{array}
\right.
\]
Search space for sane guards

Resilience condition: \( n > 3t > 0 \)

Threshold: \( 0 \leq \alpha n + \beta t + \gamma \leq n \)

\[ \begin{align*}
0 & \leq \alpha & & \leq 1 \\
-4 & \leq \beta & & \leq 4 \\
-8 & \leq \gamma & & \leq 8
\end{align*} \]

Theorem

Assume: \( n > \sum_{1 \leq i \leq k} \delta_i \cdot t_i \) and \( \forall i. t_i \geq 0 \) — resilience cond.

Threshold: \( 0 \leq \alpha n + \sum_{1 \leq i \leq k} \beta_i \cdot t_i + \gamma \leq n \) — threshold

Then:

\[ \begin{cases}
0 & \leq \alpha \leq 1 \\
-B_i & \leq \beta_i \leq B_i \quad \text{for } B_i = \delta_i + 1 \quad \text{and } 1 \leq i \leq k \\
-C & \leq \gamma \leq C \quad \text{for } C = k + 1 + 2(\delta_1 + \cdots + \delta_k)
\end{cases} \]
From $\mathbb{Q}$ to a finite search space

Synthesizing thresholds of the form $\frac{n}{2}$ or $\frac{2n}{3}$:

Assume:

Resilience condition: $n > 3t \geq 0$

Threshold: $0 \leq \frac{?'_a}{6} n + \frac{?'_b}{6} t + \frac{?'_c}{6} \leq n$

Then

$0 \leq ?'_a \leq 6 \cdot 1$

$-6 \cdot 4 \leq ?'_b \leq 6 \cdot 4$

$-6 \cdot 8 \leq ?'_c \leq 6 \cdot 8$

$?'_a, ?'_b, ?'_c \in \mathbb{Z}$
Explicit enumeration?

coefficients \((\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)\) in reliable broadcast:

\[(2 \cdot 9 \cdot 17)^2\] vectors

coefficients \((0, \theta_2, 1, 0, \theta_5, 1)\) in reliable broadcast:

\[9 \cdot 9\] vectors

coefficients \((\theta_1, \theta_2, \ldots, \theta_9)\) in one-step consensus (BOSCO):

\[(3 \cdot 17 \cdot 33)^3\] vectors

The generator should learn from the counterexamples!
Sketch of reliable broadcast in 2D

missing coefficients for \( t \):
\[
\tau_1 = ?_2 \cdot t + 1 \\
\tau_2 = ?_5 \cdot t + 1
\]

bounded search space
Counterexample to unforgeability

Model checker flags an error:

\[ n = 4, \ t = 1, \text{ and } f = 1 \]

State 1. \( \kappa_{V0} = 3 \), other counters 0

\[ \downarrow r_3 : 0 + 1 \geq 0 \cdot 1 + 1 \]

State 2. \( \kappa_{SENT} = 1 \), \( x = 1 \), \( \kappa_{V0} = 2 \)

\[ \downarrow r_5 : 1 + 1 \geq 0 \cdot 1 + 1 \]

State 3. \( \kappa_{ACCEPT} = 1 \), \( \kappa_{SENT} = 0 \)

?_2 = 0 and ?_5 = 0
Counterexample to unforgeability

Model checker flags an error:

\[ n = 4, \ t = 1, \text{ and } f = 1 \]

State 1. \( \kappa_{V_0} = 3 \), other counters 0

\[ \downarrow \quad r_3 : 0 + 1 \geq \ ?_2 \cdot 1 + 1 \]
\[ \downarrow \quad r_3 : 0 + 1 \geq \ ?_2 \cdot 1 + 1 \]

State 2. \( \kappa_{SENT} = 1 \), \( x = 1 \), \( \kappa_{V_0} = 2 \)

\[ \downarrow \quad r_5 : 1 + 1 \geq \ ?_5 \cdot 1 + 1 \]
\[ \downarrow \quad r_5 : 1 + 1 \geq \ ?_5 \cdot 1 + 1 \]

State 3. \( \kappa_{ACCEPT} = 1 \), \( \kappa_{SENT} = 0 \)

\[ ?_2 = 0 \text{ and } ?_5 = 0 \]
Counterexample to unforgeability

Model checker flags an error:

\[ n = 4, \quad t = 1, \quad \text{and} \quad f = 1 \]

State 1. \( \kappa_{V0} = 3 \), other counters 0

\[ \downarrow r_3 : 0 + 1 \geq 0 \cdot 1 + 1 \]
\[ \downarrow r_3 : 0 + 1 \geq \delta_2 \cdot 1 + 1 \]

State 2. \( \kappa_{SENT} = 1 \), \( x = 1 \), \( \kappa_{V0} = 2 \)

\[ \downarrow r_5 : 1 + 1 \geq 0 \cdot 1 + 1 \]
\[ \downarrow r_5 : 1 + 1 \geq \delta_5 \cdot 1 + 1 \]

State 3. \( \kappa_{ACCEPT} = 1 \), \( \kappa_{SENT} = 0 \)

\[ \delta_2 = 0 \quad \text{and} \quad \delta_5 = 0 \]
Learning from counterexamples

1. unforgeability

2. sanity

3. correctness

4. sanity

5. relay

6. relay

found the solution ?2 = 1 and ?5 = 2
Synthesis loop

Find \(?_1, \ldots, ?_k \in \mathbb{Q}\)
Experiments
We have synthesized reliable broadcast, hybrid broadcast, and BOSCO (one-step consensus) using forsyte.at/software/bymc.
local $myval_i \in \{0, 1\}$

while true do
  if $myval_i = 1$ and not sent ECHO before then send ECHO to all

  if received ECHO from at least $t + 1$ distinct processes and not sent ECHO before then send ECHO to all

  if received ECHO from at least $n - t$ distinct processes then accept

od

resilience: $n > 3t$, $f \leq t$ Byzantine faults
Thresholds for Byzantine reliable broadcast

local $myval_i \in \{0, 1\}$

while true do
  if $myval_i = 1$ and not sent ECHO before
  then send ECHO to all

  if received ECHO from at least distinct processes
    and not sent ECHO before
  then send ECHO to all

  if received ECHO from at least distinct processes
  then accept

od

resilience: $n \geq 3t$, $t \leq t$ Byzantine faults
Thresholds for hybrid reliable broadcast

\[
\text{local } \text{myval}_i \in \{0, 1\}
\]

while true do
  if myval\_i = 1 and not sent ECHO before then send ECHO to all
  if received ECHO from at least \(t_b + 1\) distinct processes and not sent ECHO before then send ECHO to all
  if received ECHO from at least \(n - t_b - t_c\) distinct processes then accept
od

resilience: \(n > 3t_b + 2t_c\), \(f_b \leq t_b\) Byzantine and \(f_c \leq t_c\) crash faults
Thresholds for hybrid reliable broadcast

\[ \text{local } myval_i \in \{0, 1\} \]

\begin{itemize}
  \item \textbf{while true do}
    \begin{itemize}
      \item \textbf{if } myval_i = 1 \textbf{ and not sent ECHO before}
      \item \textbf{then send ECHO to all}
    \end{itemize}
  \item \textbf{if received ECHO from at least } t + 1 \textbf{ distinct processes}
    \begin{itemize}
      \item \textbf{and not sent ECHO before}
    \end{itemize}
    \begin{itemize}
      \item \textbf{then send ECHO to all}
    \end{itemize}
  \item \textbf{if received ECHO from at least } n - t \textbf{ distinct processes}
  \item \textbf{then accept}
\end{itemize}
\textbf{od}

resilience: \[ n > 3t_b + t_c \]

\[ f_b \leq t_b \text{ Byzantine and } f_c \leq t_c \text{ crash faults} \]
Reliable broadcast: changing specifications

unforgeability: if no correct process received $\text{bcast}$, then no correct process ever accepts $\text{bcast}$

correctness and relay like before

<table>
<thead>
<tr>
<th>$X = 2$</th>
<th>$X = t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>no solutions for $n &gt; 3t$</td>
<td>no solutions for $n &gt; 3t$</td>
</tr>
<tr>
<td>3 solutions for $n &gt; 3t + 2$</td>
<td>3 solutions for $n &gt; 4t$</td>
</tr>
</tbody>
</table>

each answer found in less than $12$ sec.
BOSCO (one-step consensus)

\[ \phi_A \land s_1 < \tau_D \land s_1 < \tau_D \land s_0 + f \geq \tau_U \land s_1 + f \geq \tau_U \]

\[ \phi_A \land s_1 < \tau_D \land s_1 < \tau_D \land s_0 < \tau_U \land s_1 < \tau_U \]

\[ \phi_A \land s_1 < \tau_D \land s_1 < \tau_D \land s_0 < \tau_U \land s_1 < \tau_U \]

\[ \phi_A \land s_1 < \tau_D \land s_1 < \tau_D \land s_0 < \tau_U \land s_1 < \tau_U \]
BOSCO: synthesis results

\[
\begin{align*}
\text{Agreement} & \quad \text{when } n > 3t \\
\text{Termination} & \quad \text{when } n > 3t \\
\text{One step} & \quad \text{when } n > 7t \\
\text{Fast termination} & \quad \text{when } n > 7t \quad \text{or} \quad n > 5t, f = 0
\end{align*}
\]

Found 4 solutions using 4 cluster nodes = 64 cores \hspace{1cm} 24 min.

No solutions for \( n \geq 5t, f = 0 \) and \( n \geq 7t \) \hspace{1cm} 40 min.

(the conditions \( n > 5t \) and \( n > 7 \) are tight)
Conclusions

we can verify and synthesize distributed algorithms that are:

- asynchronous and parameterized,
- subject to faults, sending to all, and
- counting messages and comparing to threshold guards

next steps:

- learning from positive examples,
- geometrical structure of learned regions,
- synthesizing threshold automata, not just thresholds

[ forsyte.at/software/bymc ]