Consensus Beyond Thresholds: Generalized Byzantine Quorums Made Live

Orestis Alpos
University of Bern

Christian Cachin
University of Bern

Symposium on Reliable Distributed Systems 2020
September 2020
Consensus

- **Agree** on a common value
- Trust assumptions in **thresholds**
- $n = 7$
- $f = 2$
- All participants trusted **equally**
Consensus algorithms still in monoculture

- Agree on a common value
- Trust assumptions in thresholds
- $n = 7$
- $f = 2$
- All participants trusted equally
- Participants are of the same type
The participants are diverse

- Operating system
- Hardware
- Administrators
- Location

- Fail with different probabilities

- Failures are correlated

- Expressive and resilient through complex and correlated trust assumptions
Byzantine quorum systems
A rich and expressive abstraction

- $n$ parties: $\mathcal{P} = \{p_1, \ldots, p_n\}$

- Fail-prone system $\mathcal{F}$: A fail-prone set in $\mathcal{F}$ contains all the failed parties

- Quorum System $\mathcal{Q}$: The set of quorums

- Such that the Consistency and Availability conditions hold:
  \begin{align*}
  \forall Q_1, Q_2 \in \mathcal{Q}, \forall F \in \mathcal{F} : & \quad Q_1 \cap Q_2 \not\subseteq F \\
  \forall F \in \mathcal{F} : & \quad \exists Q \in \mathcal{Q} : F \cap Q = \emptyset
  \end{align*}

- By definition generalized
A threshold Byzantine quorum system

- $\mathcal{P} = \{p_1, \ldots, p_7\}$
- $\mathcal{F}:$ All subsets with cardinality $f = 2$
- $\mathcal{Q}:$ All subsets with cardinality $n - f = 5$
- Consistency
  - $|Q_1 \cap Q_2| \geq 3$, $|F| = 2$
- Availability
  - Any 2 fail, the other 5 are correct
- $n > 3f$
We need to do better

- Generalized BQS → realistic, better resilience, but not yet **practical**

- **Example**: The 2-layered-1-common generalized BQS
Related work

- Stellar consensus protocol
  - Generalized trust assumptions
  - Different for each user
  - Not based on the classical Byzantine quorum system theory
  
- Benaloh and Leichter [BL88] first secret sharing over generalized structures

- Hirt and Maurer [HM00] multiparty computation with generalized failure patterns

- Cramer, Damgård, and Maurer [CDM00] use monotone span programs for generalized multiparty computation
Implementing generalized Byzantine quorum systems

- Challenges (and a solution that would not work).
- Generalized BQS as monotone boolean formulas.
- Generalized BQS as monotone span programs.
Implementing a generalized BQS is a challenging task

Implement BQS as enumeration of all quorums 792 quorums

• Specify in user-friendly way

• Efficient and compact encoding

• Efficient quorum-checking
Generalized Byzantine quorum systems as monotone boolean formulas (MBF)
Parsing a BQS as an MBF

- Using logical **and**, **or**, **threshold** operators $\Theta_k^m(q_1, \ldots, q_m)$

```json
{
    "select": 3,  "out-of": [
        {
            "select": 2,  "out-of": ["A0", \{
                "select": 2,  "out-of": ["B0", "B1", "B2", "B3"]\}]},
        {
            "select": 2,  "out-of": ["A1", \{
                "select": 2,  "out-of": ["B3", "B4", "B5", "B6"]\}]},
        {
            "select": 2,  "out-of": ["A2", \{
                "select": 2,  "out-of": ["B6", "B7", "B8", "B9"]\}]},
        {
            "select": 2,  "out-of": ["A3", \{
                "select": 2,  "out-of": ["B9", "B10", "B11", "B1"]\}]}
    ]
}
```
Storing the BQS as an MBF

- As a tree
- size is $O(n)$, where $n$ the size of MBF
Checking for quorums

- Check whether set $A$ is a quorum
- evaluate formula on input $A$, time $O(n)$

1: $\text{eval}(F, A)$
2: \hspace{1em} if $F$ is a literal then
3: \hspace{2em} return $(F \in A)$
4: \hspace{1em} else
5: \hspace{2em} write $F = \text{op}(F_1, \ldots, F_m)$, where $\text{op} \in \{\land, \lor, \Theta\}$
6: \hspace{2em} for each $F_i$ do
7: \hspace{3em} $x_i \leftarrow \text{eval}(F_i, A)$
8: \hspace{2em} return $\text{op}(x_1, \ldots, x_m)$
Generalized Byzantine quorum systems as monotone span programs (MSP)
Monotone span programs (MSP)

- Each participant gets one vector (or more)

- If the vectors of a set of participants span a target vector, the set is accepted

- An MSP implements a quorum system if it accepts exactly its quorums

- There are functions efficiently encoded by an MSP, but not by a formula [BGW99]
Parsing a BQS as an MSP

• Insertion: \( Q_3 = Q_1(p_z \rightarrow Q_2) \)
  - \( Q_1 \) defined on \( P_1 \), \( p_z \in P_1 \)
  - \( Q_2 \) defined on \( P_2 \)
  - \( Q_3 \) replaces \( p_z \) by quorums in \( Q_2 \)

Nikon, Nikova [NN2004]

• Insertion on MSPs: \( M_3 = M_1(r_z \rightarrow M_2) \)
  - \( M_1 \) implements \( Q_1 \)
  - \( M_2 \) implements \( Q_2 \)
  - \( M_3 \) can be constructed to implement \( Q_3 \)

• Given a formula, create the MSP with recursive insertions of nested sub-formulas
Parsing a BQS as an MSP

- Construct the MSP that implements a given MBF
  - Recursive insertions.
  - The Vandermonde matrix $V(n,t)$, when seen as an MSP, implements the access structure $\Theta^n_t(q_1, \ldots, q_n)$

$$V(n, t) = \begin{pmatrix}
1 & x_1 & x_1^2 & \cdots & x_1^{t-1} \\
1 & x_2 & x_2^2 & \cdots & x_2^{t-1} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & x_n & x_n^2 & \cdots & x_n^{t-1}
\end{pmatrix}$$

$x_i \neq x_j \neq 0$, for $1 \leq i \leq j \leq n$
Checking for quorums using an MSP

- Check whether $M_A^T x = e_1$ has solutions, using Gaussian elimination.

- Time complexity is $O(n^3)$, where $n$ the dimension of $M$, can be optimized using PLU-decomposition (but still cubic on average).
Consensus beyond thresholds: Generalized HotStuff
HotStuff

- Consensus algorithm by Yin et al [YMRGA19]
- Efficient, linear communication, speed of network
- Libra cryptocurrency
- Replicas run the protocol
- Clients submit commands and collect responses
Generalized HotStuff

• Protocol advances in epochs

• Each epoch four phases

• In each phase
  – The leader creates a proposal and sends to other replicas
  – The replicas validate and vote
  – The leader waits for \( n - f \) quorum of votes
  – Upon receiving them, creates a certificate, used in next proposal

• The generalized protocol satisfies the same safety and liveness properties as threshold HotStuff
## Evaluated systems

<table>
<thead>
<tr>
<th>System</th>
<th>BQS implementation in</th>
<th>Supported types of BQS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>replicas</td>
<td>clients</td>
</tr>
<tr>
<td>Counting-All</td>
<td>counting</td>
<td>counting</td>
</tr>
<tr>
<td>MBF-All</td>
<td>MBF</td>
<td>MBF</td>
</tr>
<tr>
<td>MSP-All</td>
<td>MSP</td>
<td>MSP</td>
</tr>
<tr>
<td>MSP-Replicas</td>
<td>MSP</td>
<td>-</td>
</tr>
</tbody>
</table>

- Based on the *prototype HotStuff* implementation: github.com/hot-stuff/libhotstuff
When the number of parties is small all the generalized protocols are efficient

- 4 replicas, varying number of clients (1 up to 8) and request rate
- All systems instantiated with a threshold BQS with $n = 4, f = 1$
In larger systems the **MBF-All** protocol is as efficient as the original **Counting-All**.

The **MSP-Replicas** protocol is still comparable to **Counting-All**.

- Up to 31 replicas and 32 clients
- All systems again instantiated with a threshold BQS with $n = 3f + 1$
The complexity of the BQS moderately affects the MBF-All implementation

- 16 up to 40 replicas and 32 clients
- MBF-All instantiated with the threshold and the 2L1C BQS
The complexity of the BQS affects the **MSP-Replicas** protocol only slightly

- 16 up to 40 replicas and 32 clients
- **MSP-Replicas** instantiated with the threshold and the 2L1C BQS
The **MBF-All** protocol outperforms the **MSP-Replicas**

- 16 up to 40 replicas and 32 clients
- Systems instantiated with the threshold and the 2L1C BQS
Thank you!

Full paper: arxiv.org/abs/2006.04616

Blogpost: cryptobern.github.io/beyondthreshold/

Orestis Alpos
orestis.alpos@inf.unibe.ch
crypto.unibe.ch/oa/
Twitter: @alpenliebious

[MR98] DBLP:journals/dc/MalkhiR98
[BGW99] DBLP:journals/combinatorica/BabaiGW99
[NN04] DBLP:journals/iacr/NikovN04
[YMRGA19] DBLP:conf/podc/YinMRGA19
[BL88] DBLP:conf/crypto/Leichter88
[HM00] DBLP:journals/joc/HirtM00
[CDM00] DBLP:conf/eurocrypt/CramerDM00