Fever: Optimal Responsive View Synchronisation via Responsive Asynchronous Clock Synchronization

ITTAI ABRAHAM AND ANDREW LEWIS-PYE

Hotstuff showed how to achieve optimistic responsiveness while keeping message complexity within each view $O(n)$, but left open the question as to how to achieve $O(n)$ complexity for *view changes*. Here, we describe a simple method that is optimistically responsive and gives $O(n)$ complexity for view changes in the worst case, i.e. not just in expectation. The synchronisation technique can be applied to other leader based protocols.

1 THE SETTING

We use a standard setup:

- We consider a set of n processors with full PKI and a threshold signature scheme. Some unknown subset of f many processors may display Byzantine faults, where $n \geq 3f + 1$.
- Each pair of processors has an authenticated communication channel between them, i.e. communication is point-to-point.
- Communication is partially synchronous, i.e. there is some known bound Δ and some unknown 'global stabilisation time' GST such that, if a message is sent at time t, then it is received by time $\max\{t, \text{GST}\} + \Delta$.
- All processors have the same clock speeds and start at the same time with clocks set to 0. Both of these assumptions can be relaxed, but that is outside the scope of this note.

The underlying protocol. We suppose view synchronisation is required for some underlying protocol (such as Hotstuff) with the following properties:

- Instructions are divided into views. Each view v has a designated *leader*, denoted $\texttt{lead}(v)$.
- The successful completion of a view is marked by all processors receiving a *Quorum Certificate* (QC) for view v. The QC is a threshold signature of length $O(1)$ combining $n - f$ signatures from different processors testifying that they have completed the instructions for the view. In a chained implementation of Hotstuff, for example, the leader will propose a block, processors will send votes for the block to the leader, who will then combine those votes into a QC and send this to all processors. Note that the production of QCs is not a restrictive assumption, since if it is not satisfied one can easily amend the instructions of the protocol so that it is.
- **Sufficient time for view completion**. We suppose there exists Γ such that if $\texttt{lead}(v)$ is non-faulty, if $t \geq \text{GST}$, and if each non-faulty processor is in view v from time t until either it receives a QC for view v or until $t + \Gamma$, then all non-faulty processors will receive a QC for view v before $t + \Gamma$.

The view synchronisation task. For Γ as above, we must ensure there exists some non-faulty $\texttt{lead}(v)$ and $t \geq \text{GST}$ such that each non-faulty processor is in view v from time t until either it receives a QC for view v or until $t + \Gamma$.

The requirement for optimistic responsiveness. Of course, we could do nothing and all clocks would always be synchronised, since they start as such. This allows one to give a trivial solution to the view synchronisation problem. For example, we could specify that each non-faulty processor is in view v during the interval $[\Gamma v, \Gamma(v + 1))$.

What we want, however, is a protocol which is *optimistically responsive*. Roughly, this means that it is able to proceed at network speed, so long as leaders follow the protocol. More formally,

Emails. Andrew Lewis-Pye: a.lewis7@lse.ac.uk, andy@lewis-pye.com.

suppose all non-faulty processors are in view v at $t_0 \geq$ GST, and consider any non-empty interval $[t_0, t_1]$. Let V be the set of views v' such that some non-faulty processor is in view v' during $[t_0, t_1]$, and suppose that $\texttt{lead}(v')$ is non-faulty for every $v' \in V$. Being optimistically responsive means that for every $x \in \mathbb{N}$ there exists $c \in \mathbb{N}_{>0}$ such that, if every message sent by a non-faulty processor while in any view in V is received within time Δ/c, then non-faulty leaders will produce at least x many QCs for views in V during $[t_0, t_1]$.

Being optimistically responsive within views. To produce an optimistically responsive protocol, we have to assume that the underlying protocol is optimistically responsive *within views*, which means the following. Suppose $\texttt{lead}(v)$ is non-faulty and $t \geq$ GST. For every $\Gamma' > 0$ there exists $c \in \mathbb{N}_{>0}$ such that, if each non-faulty processor is in view v from time t until either it receives a QC for view v or until $t + \Gamma'$, and if all messages sent by non-faulty processors while in view v are received within time Δ/c, then all non-faulty processors will receive a QC for view v before $t + \Gamma'$.

2 RESPONSIVE ASYNCHRONOUS CLOCK SYNCHRONIZATION

Clock-times. To synchronise processors while maintaining optimistic responsiveness, we have a predetermined 'clock-time' corresponding to each view: The clock-time corresponding to view v is $t_v := \Gamma v$. At certain points in the execution, a processor may instantaneously forward their clock to some clock-time t_v and enter view v.

View Certificates. When a non-faulty processor enters a view v such that $\texttt{lead}(v) \neq \texttt{lead}(v-1)$, it will send a `view` v message to $\texttt{lead}(v)$. Once $\texttt{lead}(v)$ receives $f + 1$ `view` v messages, it will combine these into a single threshold signature, which is a view certificate (VC) for view v, and will send this VC to all processors.[1]

The safety condition. We want to ensure that when processors forward their clocks a certain *safety condition* is maintained. To define the condition, let $t(p)$ denote the value of processor p's clock. At any point e in an execution, let $T(e) := \{t(p) : \ p \text{ is non-faulty}\}$. Our safety condition is:

(‡) Let e be any point in an execution. For any $t \in T(e)$:

$$|\{t' \in T(e) : \ t' \geq t - \Gamma\}| \geq f + 1.$$

The motivation behind maintaining this condition is that it is precisely what we will need to establish view synchronisation.

When processors forward clocks. At any point in the execution, if a non-faulty processor receives a QC for view $v - 1$ or a VC for view v, and if $t(p) < t_v$, then they will instantaneously forward their clock to t_v. We forward clocks upon receiving a QC to achieve optimistic responsiveness. We forward clocks upon seeing a VC to achieve view synchronisation. Since these are the only conditions under which non-faulty processors forward their clocks, it is already easy to argue that the safety condition (‡) is maintained.

Proving that (‡) always holds. To prove the claim we use a fact that will be apparent once we have defined the view synchronisation protocol precisely: (†) At any point in an execution, if non-faulty p has already contributed to a QC for a VC for view v, then $t(p) \geq t_v$. To prove (‡) always holds using this fact, suppose towards a contradiction that there is a first point of the execution, e say,

[1] It is convenient throughout to assume that when a leader sends a message to all processors, this includes itself.

for which there exists some non-faulty processor p such that $|\{t \in T(e) : \ t \geq t(p) - \Gamma\}| < f + 1$. Then p must forward its clock at e. There are two possibilities:

(1) p forwards its clock because it receives a VC for some view v with $t_v > t(p)$. In this case, there must exist at least one non-faulty processor $p' \neq p$ which contributed to the VC for view v. By the choice of e, when p' contributed to the VC we had $|\{t \in T(e) : t \geq t(p') - \Gamma\}| \geq f + 1$. Since $t(p') \geq t_v$ when it contributed to the VC, and since $t(p) = t_v$ at e, at e we have that $|\{t \in T(e) : t \geq t(p) - \Gamma\}| \geq f + 1$ also, which gives the required contradiction.
(2) p forwards its clock because it sees a QC. In this case, at least $f + 1$ non-faulty processors must have contributed to the QC, which directly gives the required contradiction.

3 THE VIEW SYNCHRONISATION PROTOCOL

The need for views of two types. The simplest approach would be to have all processors begin in view 0, to set $\texttt{lead}(v)$ to be processor v mod n, and to have a non-faulty processor p enter view v as soon as $t(p) = t_v$. The problem with this approach is that it does not quite satisfy view synchronisation, because (‡) can only be maintained with respect to Γ, which is the maximum time allocated to each view. If a non-faulty processor p is the first to enter view v, and if p enters v after GST, then (‡) ensures f other non-faulty processors will enter view v within time Γ. If the leader is non-faulty, they will form a VC for view v, which will be received by all non-faulty processors within time $\Gamma + 2\Delta$ of p entering view v, but by this time p will already have entered view $v+1$.

The solution. The solution is simple. For some parameter $k \geq 3$ (which can be chosen to suit the protocol designer's needs), we proceed as follows:

- Views are grouped into sets of k, so that the leader for view v is processor $\lfloor v/k \rfloor$ mod n. If v mod $k = 0$, then v is called 'initial'.
- If v is initial, then a non-faulty processor p enters view v when $t(p) = t_v$.
- If v is not initial, then a non-faulty processor enters view v if they are presently in a lower view and they receive a QC for view $v - 1$. Upon entering v, p forwards its clock to t_v if $t(p) < t_v$, and leaves $t(p)$ unchanged otherwise.

Summarising the view synchronisation protocol:

- **Clock-times**. The clock-time corresponding to any view v is $t_v := v\Gamma$, where Γ is as specified in Section 1.
- **Initial views**. For some parameter $k \geq 3$, $\texttt{lead}(v)$ is processor $\lfloor v/k \rfloor$ mod n. If v mod $k = 0$, then v is called 'initial'.
- **When processors forward clocks**. At any point in the execution, if a non-faulty processor receives a QC for view $v - 1$ or a VC for view v, and if $t(p) < t_v$, then they instantaneously forward their clock to t_v.
- **When processors enter views**. If v is initial, then p enters view v when[2] $t(p) = t_v$, and p sends a $\texttt{view}\ v$ message to $\texttt{lead}(v)$ at this point. If v is not initial, then p enters view v if it is presently in a view $< v$ and it receives a QC for view $v - 1$.
- **The formation of VCs**. At any point in the execution, if $\texttt{lead}(v)$ is in some view $\leq v$ and receives at least $f + 1$ $\texttt{view}\ v$ messages, they form a VC for view v and send this to all processors.

[2]So, all processors start in view 0.

4 PROVING VIEW SYNCHRONISATION

For the sake of simplicity, we assume $\Gamma > 2\Delta$, but a similar argument will apply for smaller Γ (using larger k). Suppose v is initial, $\texttt{lead}(v)$ is non-faulty and that the first time e_0 at which any non-faulty processor p_0 enters view v satisfies $e_0 \geq GST$. Note that a non-faulty processor cannot enter any view $v' > 0$ unless some non-faulty processor has already entered $v' - 1$ or else another non-faulty processor has already entered view v'. This means no non-faulty processor has entered any view $v' > v$ at e_0. Let e_1 be the first time that a non-faulty processor p_1 enters the next initial view, i.e. view $v + k$. In the below, we prove more than the fact that $\texttt{lead}(v)$ will produce a QC for one of the views in $[v, v + k)$. We show that $\texttt{lead}(v)$ will produce QCs for multiple successive views if k is large enough, since this will be useful in some implementations (such as chained implementations of Hotstuff etc).

(1) Suppose first that p_1 does not forward their clock while in views $v, \ldots, v + k - 1$. Let $e_0' \geq e_0$ be the time at which p_1 enters view v. Then $e_1 = e_0' + k\Gamma \geq e_0 + k\Gamma$. Since (‡) is satisfied when p_0 enters view v at e_0, at least $f + 1$ $\texttt{view}\ v$ messages will have been sent to $\texttt{lead}(v)$ by $e_0 + \Gamma$. Since $\Gamma > 2\Delta$ and $k > 1$, $\texttt{lead}(v)$ will have sent out a VC for view v by $e_0 + \Gamma + \Delta$ and all non-faulty processors will enter view v by $e_0 + \Gamma + 2\Delta$. Then, for each $k' \in \mathbb{N}_{\geq 0}$ such that $(k' + 2)\Gamma + 2\Delta < k\Gamma$, all processors will receive a view $v + k'$ QC by $e_0 + (k' + 2)\Gamma + 2\Delta$. Since $\Gamma > 2\Delta$, this means $\texttt{lead}(v)$ will produce a QC for (at least) all views in $[v, v + k)$ except at most the last two. This set is non-empty, since $k \geq 3$.

(2) Suppose next that the previous case does not apply, and let k' be the greatest such that p_1 forwards their clock because they see a QC for view $v + k'$. This already means that $\texttt{lead}(v)$ produces a QC for some view in $[v, v + k)$, but we want to show that in this case $\texttt{lead}(v)$ produces a QC for all these views except at most the last. If $\texttt{lead}(v)$ produces QCs for either of views $k - 1$ or $k - 2$ we are done, so suppose otherwise and that $k' \leq k - 3$. Let e be the time at which p_1 forwards their clock because they see a QC for view $v + k'$. This means p_1 enters the next initial view $v + k$ at a time $\geq e + 2\Gamma$. All non-faulty processors receive a QC for view $v + k'$ by time $e + \Delta$, and will be in view $v + k' + 1$ by this time. Then p_1 will receive a QC for view $v + k' + 1$ by time $e + \Delta + \Gamma$, which gives the required contradiction, since $\Delta < \Gamma$.

Establishing optimistic responsiveness. This is just a matter of checking the definition. Suppose all non-faulty processors are in view v at $t_0 \geq$ GST, and consider any non-empty interval $[t_0, t_1]$. Let V be the set of views v' such that some non-faulty processor is in view v' during $[t_0, t_1]$, and suppose that $\texttt{lead}(v')$ is non-faulty for every $v' \in V$. It is then clear from the definition of the protocol (and since the underlying protocol is optimistically responsive within views) that for any $x \in \mathbb{N}$ there exists $c \in \mathbb{N}$ such that, if every message sent by a non-faulty processor while in any view in V is received within time Δ/c, then non-faulty leaders will produce at least x many QCs for views in V during $[t_0, t_1]$. This holds because there is no limit to how quickly QCs can be produced if messages are delivered fast enough, and because if messages are delivered fast enough then processors will not enter any view $v' \in V$ before seeing a QC for view $v' - 1$, but will enter view v' immediately upon seeing such a QC.