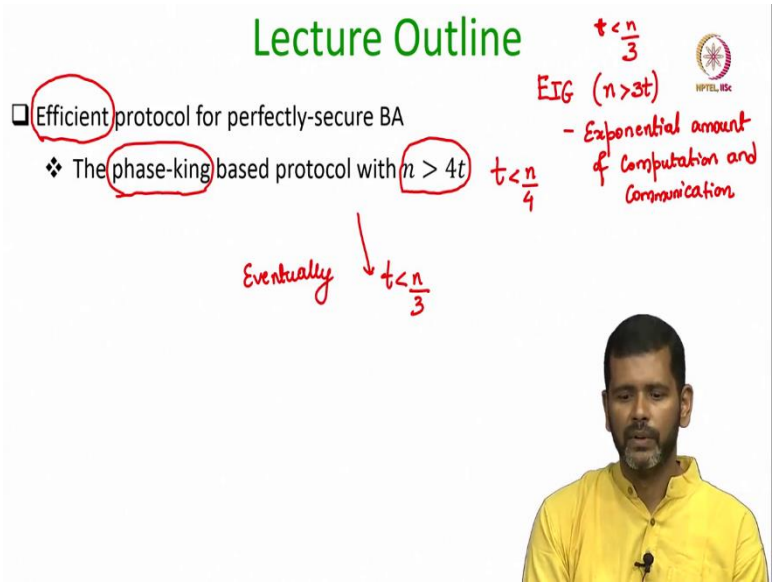
Secure Computation: Part II
Prof. Ashish Choudhury
Department of Computer Science and Engineering
Indian Institute of Science, Bengaluru

Lecture - 07
Efficient Protocols for Perfectly - Secure Byzantine Agreement: Part I

(Refer Slide Time: 00:33)



Hello everyone, welcome to this lecture. So, we will now start discussing about efficient protocols for perfectly-secure Byzantine Agreement. So, recall that till now we have seen the EIG protocol for perfectly secure byzantine agreement, that protocol was designed with the condition, and it was inefficient in the sense that, it requires exponential amount of computation and communication.

So, now we will focus on designing protocols, perfectly secure byzantine agreement and broadcast protocols, where the parties need to perform only polynomial amount of computation and communication. So, we will first see a very simple protocol, this protocol is designed with the condition$n > 4t$.

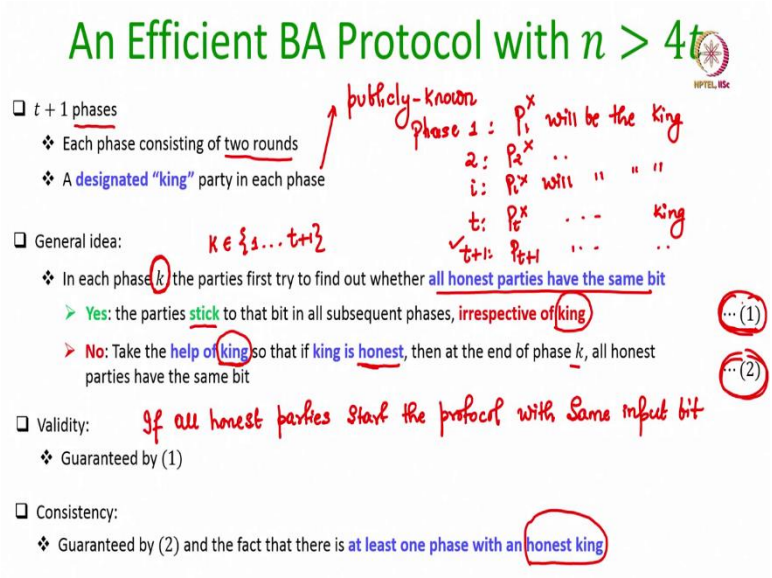
That means, while the EIG protocol can tolerate up to t corruptions where$t < \frac{n}{3}$, the protocol that we are going to discuss today can tolerate only up to $\frac{n}{4}$ number of corruptions; that means, the corruption capacity the capacity to tolerate the number of faults for the protocol that we are going to discuss today is less compared to the EIG

protocol.

But in return, we get efficiency, namely the protocol does not require the parties to perform exponential amount of computation or communication, and this protocol is based on a very nice paradigm called as the phase king paradigm. In the subsequent lectures we will see that we can improve this protocol further, in the sense that we can change the protocol and we can tolerate up to $t < \frac{n}{3}$ corruptions; that we will see in some subsequent lectures.

So, eventually we will see a better version of the phase king protocol, which can tolerate up to $\frac{n}{3}$ corruptions.

(Refer Slide Time: 02:50)



So, this protocol consists of $t + 1$ phases and each phase consists of two rounds. And there will be a designated king party for each phase which will be publicly known. So, for simplicity we can imagine that in phase 1, party P_1 will be the king in phase 2 party P_2 will be the king and in general in phase i party P_i will be the king.

But of course, the parties can follow any other assignment of the kings to the respective parties. The only thing is that we have to ensure that in each phase a designated party or fixed party is assigned as the king and the kings are never repeated; in the sense that each phase will have a unique king assigned to it and the parties will be aware of the fact which party has been assigned as the designated king for which phase.

So, the general idea behind the protocol is the following. In each phase k the parties first try to find out whether all the parties have the same bit or not. So, for that they will exchange messages with each other and there are two possibilities yes and no, if it turns out that in some phase k the all the honest parties have the same bit, then we need to ensure in the protocol that the parties stick to that bit that common bit in all the subsequent phases namely phase number $k+1, k+2, k+3$ all the way up to $t+1$.

Because what I am saying is with respect to any phase k where $1 \leq k \leq t+1$. Say for instance if in the second phase it is identified that all the honest parties have the same bit, then somehow in the protocol we need to ensure that from third phase onwards, all the honest parties stick to that bit that common bit irrespective of what exactly is the status of the king; whether the king is honest, whether the king is corrupt what kind of messages king is communicating and so on.

So, that is the first property which we will ensure in the protocol. How? - that will be clear from the protocol steps.

Whereas, in phase k, if it turns out if the parties somehow identify that all the honest parties do not have the same bit, then they take the help of the king party, so that, if the king is honest, then at end of phase k all the honest parties have the same bit ok. So, you see the role of the king comes into the picture, only when during the phase k turns out that all the honest parties do not have the same bit. If they have the same bit, the help of king will not be considered at all.

But if all the honest parties do not have the same bit during the phase k, then the help of king will be sought. Now, it could be possible that king is corrupt, in that case king might confuse different parties by communicating different messages, different bits. But what will be ensured is that if the designated king for the kth phase is honest, then using the help of the king party all the honest parties will come to an agreement at the end of phase k.

And if they come to an agreement at the end of phase k, right then when they go to the next iteration, when they go to the next phase anyhow the condition one will be satisfied. Because parties have already reached agreement at the end of phase k. So, they will be remaining in agreement at the end of phase $k+1, k+2$ all the way to phase $t+1$ due to

this property number 1.

Now, for the moment assume that both these properties are achieved somehow in the protocol, let us see that how we get the validity and the consistency properties. The validity property is guaranteed, if we ensure the first condition in the protocol.

This is because, if all honest parties, start the protocol with same input bit, then they will stick to that bit at the end of every phase, because the help of king will not be considered at all because of this first property achieved in the protocol. And at the end of $t + 1$th phase, the parties will output that common input.

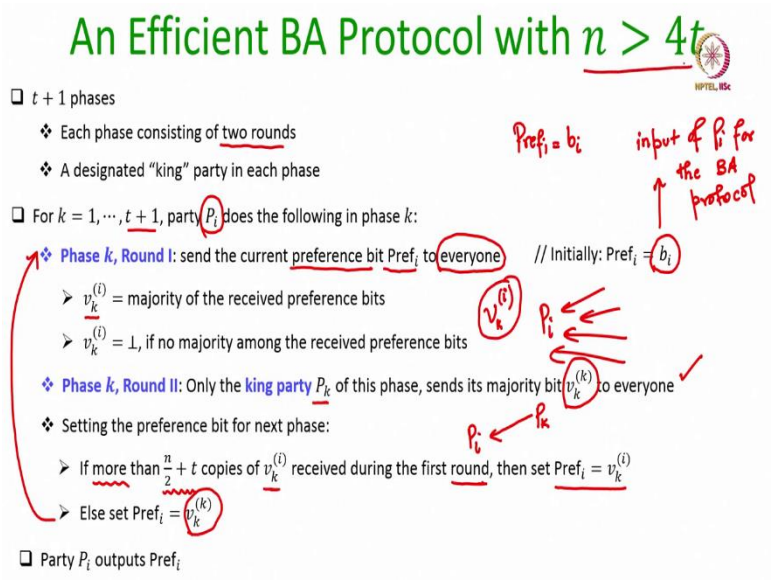
So, that ensures the validity property. The consistency property is guaranteed because we are running the protocol for $t + 1$ phases and in each phase a unique party is assigned as the king. So, in the worst case what can happen is that the first t phases might have a corrupt king. So, it might be the case that P_1 is corrupt, P_2 is corrupt, P_i is corrupt and P_t is also corrupt.

So, it might be possible that during the first t phases king is not helping at all to reach the agreement, but as soon as we reach to the $t + 1$ phase, we will have a party which is guaranteed to be honest. And assuming that property two is achieved in the protocol, the honest king during the $t + 1$th phase will ensure that at the end of the $t + 1$th phase all the honest parties have the same output.

Of course, it could be possible that the honest king, which is guaranteed to exist among the $t + 1$ kings, appears somewhere earlier. It might be appearing in say phase number 2, then we will be reaching agreement at the end of phase 2 itself. Of course, we have to run the protocol for all the $t + 1$ phases, because the parties will not be knowing the exact identity of the honest king.

Remember no one knows the identity of the bad parties and the good parties. They only know the number of parties which can be corrupt namely only the parameter t will be publicly known. So, that is the general idea behind the protocol. Now, let us see how exactly we ensure the property number 1 and property number 2 in the protocol ok.

(Refer Slide Time: 10:34)



So, here is the protocol code. So, for $k = 1 \ldots t + 1$, every party P_i does the following executes the following steps and when I say every party P_i; that means, this sequence of actions has to be performed by every party of course, if party P_i gets corrupted by the byzantine adversary, then it may not follow the instructions which I am going to discuss below.

But if the party P_i is not under the control of the adversary, it will stick to these instructions. So, there will be two rounds in each phase, in the first-round parties try to find out whether they have the same bit for this iterate, during this iteration or during this phase. So, we can imagine a phase as an iteration and in each iteration, we have two rounds. So, the first round of phase k involves the following communication, every party P_i sends its current preference bit which I denote by $Pref_i$ to everyone.

Now, you might be wondering what the initial value of $Pref_i$ is when we start this protocol namely the first phase you need to begin with every party assigns its input for the BA protocol, which is b_i. So, b_i is the input of P_i for the BA protocol. So, in the first phase the value of $Pref_i$ will be b_i.

But in general, when we go to the kth phase, in each phase the value of the preference bit will be updated based on some decision rules, and whatever is the current preference bit for the ith party, it will send to everyone including itself that is also important.

Whenever I explain a code and we have a step ``send something to everyone", until and unless it is explicitly stated, it means that the party is sending this message to itself also. Of course, you might be wondering, how can a party send something to itself, well we can imagine that logically inside it is making statement which considers that it would be receiving this value from itself.

So, it sends a preference bit to everyone of course, if the ith party is corrupt it may send different versions of Pref_i to different parties right. And at the end of the first round during the phase k what we do is the following, we assign the majority of the receive preference bit by $v_k^{(i)}$ whereas, if no majority is there among the received preference bits, then we set this value to null or $\perp$ or some garbage value; you can imagine $\perp$ is like a default value and it represents neither 0 nor 1.

So, what is happening here is that P_i will be sending its preference bit to everyone and in the same round it will be receiving preference bits from other parties, because other parties also will be following this code. So, they will be sending their preference bits to P_i and P_i will be having n preference bits at the end of the first round, it will take the majority if there is any majority that majority value is assigned to $v_k^{(i)}$ otherwise the value $\perp$ is assigned.

Now, during the second round of the phase k only the king party P_k does the communication. No other party performs any communication. So, during the second round the party P_k sends its majority bit, which it has assigned at the end of the previous round to everyone. Again, if the king is corrupt, it may send different versions of its majority bit to different parties, but if the king is honest it will stick to this protocol code and it will send an identical copy of its majority bit to everyone.

Now, there will be no more communication in this phase, the two rounds are over; the parties have to set the preference bit for the next phase. How do they set the preference bit? Each party P_i checks whether the majority bit which it has assigned. Whether that value has been received $\frac{n}{2} + t$ number of times during the first round of the kth phase right. So, remember P_i has received preference bits from several parties and based on that, it has set the value $v_k^{(i)}$, what this means is that P_i checks whether this value $v_k^{(i)}$ has been received from $\frac{n}{2} + t + 1$ number of parties at least.

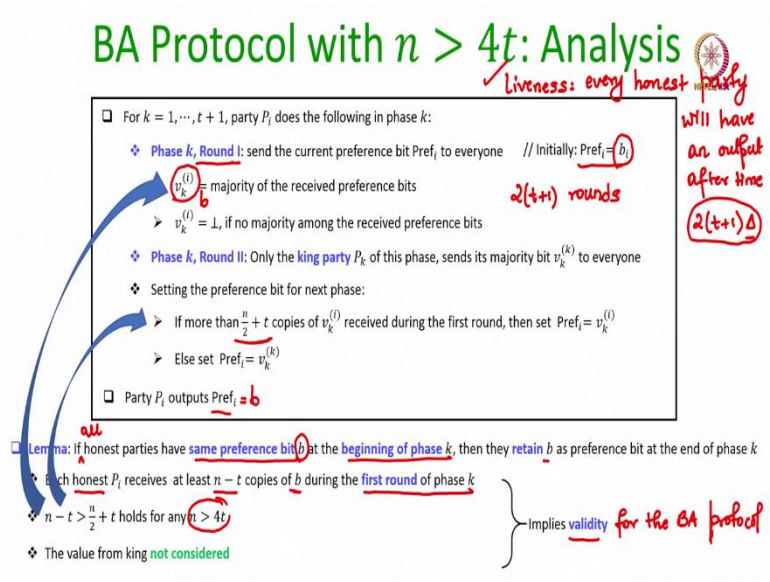
Namely, if it has received more than these many copies during the first round. If it has received more than these many copies of its majority bit, then it sets its preference bit to the majority bit. However, it could be possible that even though $v_k^{(i)}$ is set as the majority bit, it has not been received more than $\frac{n}{2} + t$ number of times, it is just a majority value that is all.

That means, if it has not been received $\frac{n}{2} + t + 1$ times, then what the party P_i does is that it sets its preference bit for the next phase to the value to the majority value which has been received from the king.

So, remember at the end of the round 2, P_i would have also received a communication from the king party P_k, where the king party would have sent its majority bit to P_i of course, if P_k is corrupt, it can send any garbage value, but if P_k is honest, then indeed the value which P_i receives during the second round of phase k will be the majority bit of the king P_k. So, P_i will set its preference bit to that value, if its own majority bit is not received more than $\frac{n}{2} + t$ number of times.

And then, the parties go to the next iteration. Once the parties execute the steps for $t + 1$ phases, at the end of $t + 1$ phases, every party outputs whatever preference bit it has. That is the output of the protocol. So, this is the BA protocol, now we must show that this protocol satisfies the liveness validity and consistency properties provided $n > 4t$. So, we will do the analysis.

(Refer Slide Time: 18:23)



So, one liveness property is trivial to verify right. So, every party every honest party will have an output, after time $2(t + 1)\Delta$. Why? Because there are total $2(t + 1)$ number of rounds, in the protocol communication rounds in the protocol and assuming that the time delay for each or the time period between time period for each round is Δ clock cycles, then after this much time, where Δ is publicly known, the parties will output some value; that means, it will not happen that the party is keep on executing the protocol forever. So, that ensures the liveness property.

Now, let us prove the validity and the consistency properties, for that we will prove some helping lemmas. So, the first helping lemma is, that if all the honest parties right if all honest parties have the same preference bit at the beginning of any phase k, then they retain the same bit as their preference bit even at the end of the phase k; that means, if already the parties have reached agreement at the beginning of the phase k in terms of their preference bits. Of course, the parties will not be knowing whether they have already reached agreement or not.

Because they will not be knowing what the preference bits of the other parties are at the beginning of any phase k. But what we are claiming here is that, if at all during the protocol execution, at the beginning of some phase k, all the honest parties have the same preference bit say b where b could be either 0 or 1, then the preference bit which the honest parties set at the end of the phase k remains the same namely b.

So, let us prove this property. So, since we are assuming that all honest parties have the same preference bit at the beginning of phase k; that means, at the end of first round of phase k. At the end of the first round of phase k, each honest party will receive at least $n - t$ copies of the value b.

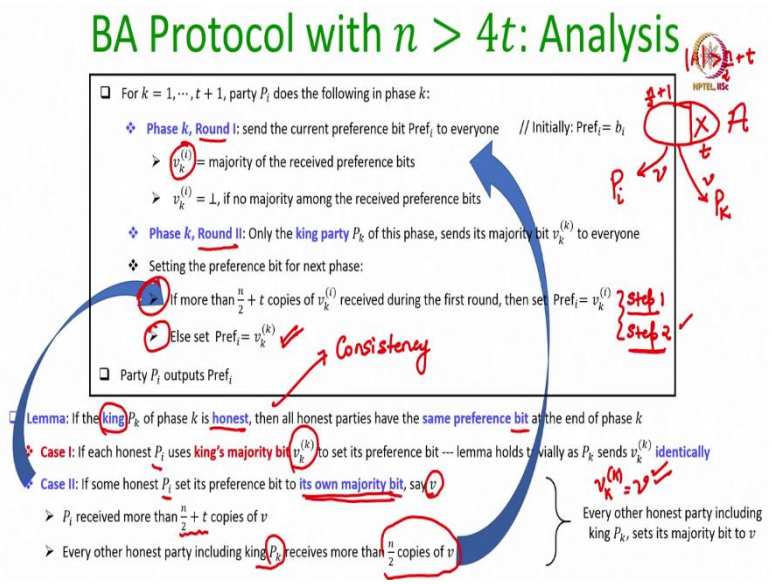
Because every honest party will say that its preference bit is the value b. So, there will be $n - t$ copies of the value bit of the value b which will be received by the party P_i every honest party P_i. And since we are assuming that $n > 4t$, then trivially $n - t$ is strictly greater than $\frac{n}{2} + t$.

That means, if we go into the protocol code, then every honest party will set its majority bit $v_k^{(i)}$ to the value b and the value from the king will not be considered at all because, every honest party would have received every honest party P_i would have received more than $\frac{n}{2} + t$ copies of the bit b, during the first round of phase k. So, that is why they will stick to the value b while setting the preference bit for the next iteration.

Now, this lemma immediately implies the validity property for the BA protocol implies validity for the BA protocol. This is because, if all the honest parties start the protocol with the same input bit; that means, at the beginning of the first iteration itself, they have the same input bit and during the first iteration, the preference bit of every party is its own input for the BA protocol; that means, what we are getting here is that at the end at the beginning of the first iteration itself, all the honest parties have the same preference bit.

So, they will retain that preference bit at the end of the phase 1; that means, when they go to phase number 2, they all have again the same preference bit, they retain that at the end of the phase 2 and like that, as and when as they keep on going to the subsequent phases, they are not going to change their preference bit, they will stick to the starting bit with which they all started the protocol and that will be the overall outcome of the protocol and that shows the validity property is satisfied.

(Refer Slide Time: 23:47)



Now, the second lemma. If the king for the phase k is honest; then the claim is that all honest parties will have the same preference bit at the end of that phase. Whichever phase an honest king is there at the end of that phase the preference bit of all the honest parties will be the same, irrespective of what messages the corrupt parties communicate during the phase k. So, there are two possible cases here, while proving this lemma, Case 1: if each honest party uses the majority bit sent by the king to set its preference bit right.

So, if you see the code, there are two possible ways through which every party sets its preference bit; either it can execute the step number 1 for setting the preference bit or it might be setting its preference bit through the step number 2.

So, the case number 1 is, when all the honest parties follow step number 2, during the protocol execution to set their preference bit; that means, no honest party receives more than $\frac{n}{2} + t$ copies of its own majority bit. If that would have been the case, everyone would set their preference bit to the majority bit sent by the king and since we are considering an honest king for this phase it will send an identical value of its own majority bit to everyone. It will not send different versions of its majority bit to different honest parties.

And since this value is said to be the preference bit for the next iteration; that means, the lemma is true ok. The other case could be when some subset of honest parties follows step number 1 to set their preference bit. While another subset of honest parties follows

step number 2 to set their preference bit during the phase k. Again, that could be possible depending upon what exactly what the initial inputs of the parties and what messages corrupt parties communicate and so on right. It is not necessary that if the king is honest and everyone is using the step number 2 executing the step number 2 to update their preference bit.

One group of honest parties might follow step number 1, one group of honest parties might follow step number 2, depending upon what exactly their state configuration is. We will show that even in this case, at the end of the phase k all the honest parties will have the same preference bit. Let us see how. So, suppose there is some honest P_i which use which executes step number 2 to update its preference bit; that means, it sets its preference bit to its own majority bit namely the value $v_k^{(i)}$.

Now, let us see the protocol code and argue that why that honest P_i would have sent its preference bit to its own majority bit it is because it would have received more than $\frac{n}{2} + t$ number of copies of this value v; whatever the majority bit it has set right; that is why it is setting its preference bit to that value v. Now, among this $\frac{n}{2} + t$ copies of the value v which P_i has received, at least $\frac{n}{2}$ copies, at least $\frac{n}{2} + 1$ copies would have come from the honest parties.

So, pictorially imagine that P_i has received the value v from this subset of parties say $\mathcal{A}$, and the cardinality of this subset of parties $\mathcal{A}$ is more than $\frac{n}{2} + t$. Maximum t parties in this set $\mathcal{A}$ could be corrupt, but more than $\frac{n}{2} + 1$ number of parties in this subset $\mathcal{A}$ would have been honest.

They would have sent the value v to other honest parties as well, including the king P_k right, during the first round of phase k, when every party is exchanging its current preference bit, P_i would have received more than $\frac{n}{2} + t$ number of copies of v.

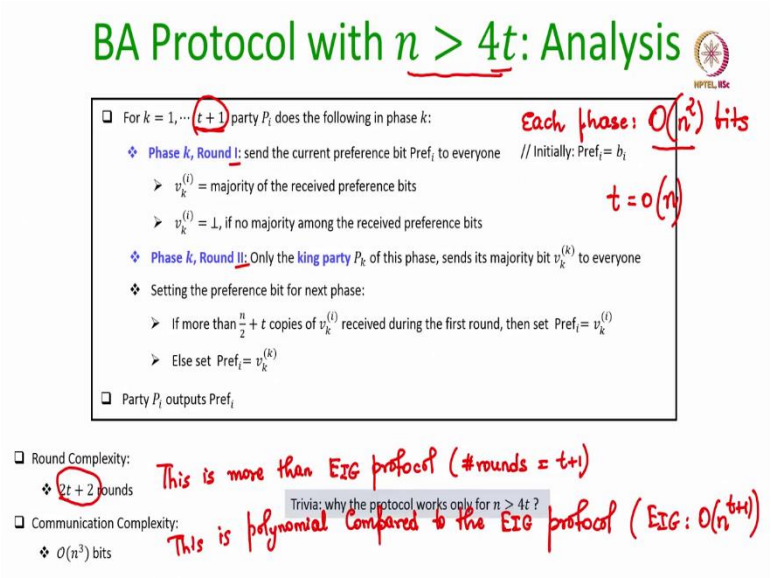
My claim is that among those $\frac{n}{2} + t + 1$ number of copies of v at least $\frac{n}{2} + 1$ number of copies of v will go to every other honest party. t corrupt parties in $\mathcal{A}$ might send different version of their preference bit to different honest parties. But still at least $\frac{n}{2} + 1$ number of copies of the value v will go to every other party including the king P_k. Consequently,

the king P_k would have sent would have set its majority bit to the value v itself, because it has received more than $\frac{n}{2}$ copies of the value v at the end of round 2.

And what is the value, which king propagates during the second round of phase k its own majority value, which is going to be v only. So, it does not matter whether any party uses king's versions of the majority bit or its own version of the majority bit it is going to be v only.

And that ensures at the end of this phase k, everyone will be on the same page in terms of their preference bits. Everyone will set their every honest party will set their preference bit to the value v of course, corrupt parties can set their preference bit to anything, we do not care about them right.

(Refer Slide Time: 30:51)



And that ensures the consistency property, because as I said among those $t + 1$ phases, there will be at least one phase where the corresponding king will be honest. It could be either phase number 1 or phase number 2 or phase number 3 or phase number $t + 1$ we do not care in which phase the designated king is honest.

But in whichever phase the king is honest, at the end of that phase the agreement will be achieved. Of course, it could be possible that in some previous phase itself, where the king was not corrupt, where the king was corrupt, but it behaves honestly; that means, even though it is corrupt by their adversary, it still sticks to the protocol code.

Well in that case, the agreement would have been achieved in some earlier phase itself we do not even have to wait for the phase where the king is guaranteed to be honest that is also a possibility and in that case from the previous lemma, what we know is that that once the agreement is achieved at the end of that particular phase, say k', then in all the subsequent phases agreement will be still maintained right, that is what we have proved in the earlier lemma.

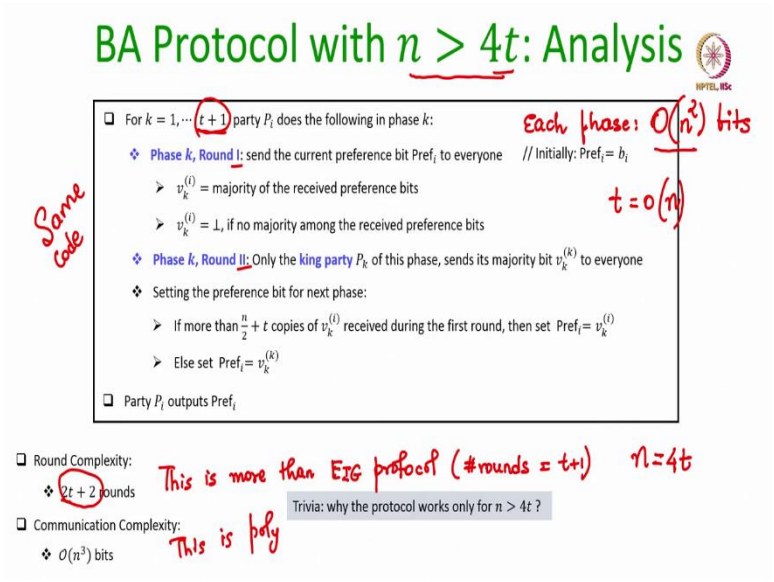
So, irrespective of whether the help of the honest king is taken or not to reach agreement or not, agreement will be achieved by the end of $t + 1$ phases that is guaranteed here. So, now, let us try to do the complexity analysis of this protocol. How many communication rounds are involved here, $2(t + 1)$rounds because there are $t + 1$ phases and in each phase, there are two communication rounds. So, total $2t + 2$ rounds are involved and this is more than EIG protocol.

So, recall that in the EIG protocol, the number of rounds was only $t + 1$. But here we need a greater number of communication rounds. So, that is one disadvantage of this protocol compared to the EIG protocol, but the good part is that the communication complexity is only $\mathcal{O}(n^3)$ bits. Why? Because each phase requires a communication of $\mathcal{O}(n^2)$ bits. Because every party needs to send its preference bit to everyone else and then the king must send its majority bit to everyone else.

So, that requires overall $\mathcal{O}(n^2)$ bits of communication in one phase and how many phases are there? There are $t + 1$ such phases. So, we can always assume that t is $\mathcal{O}(n)$. So, that is why the total communication will be $\mathcal{O}(n^3)$ bits. So, this is the good part; this is that advantage, this is polynomial compared to the EIG protocol right. Whereas, in the EIG protocol, the communication was of $\mathcal{O}(n^{t+1})$.

So, you have the trade off here. If you want to reduce the communication, but you are find to have more interaction in the protocol you can go with this protocol. Of course, another disadvantage of this protocol is it requires the condition $n > 4t$.

(Refer Slide Time: 34:56)



So, it is homework for you to go through this protocol and assume that say $n = 4t$ and run the same code and see what happens. Run the same code and see whether you have the validity and the consistency properties achieved in the modified protocol you will see that it will not be the case.

(Refer Slide Time: 35:23)

So, the protocol that I have discussed in today's lecture is taken from this textbook.

Thank you.