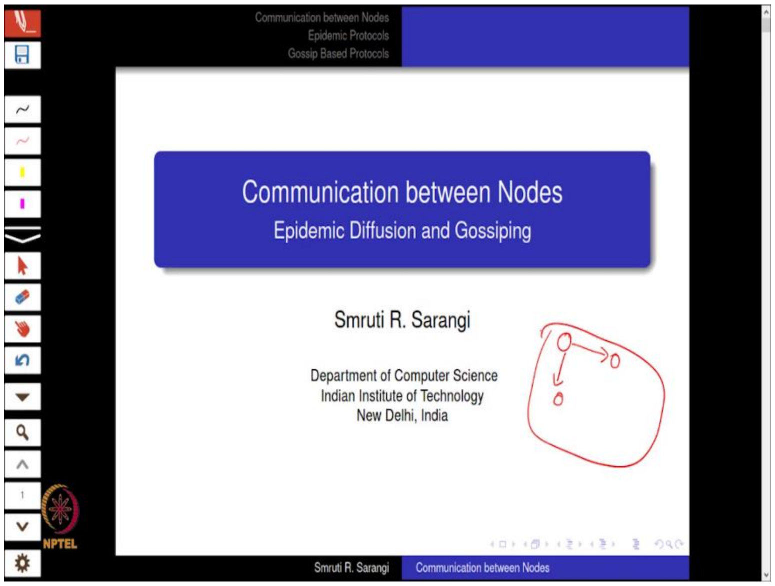
Advanced Distributed Systems

Professor Smruti R Sarangi
Department of Computer Science
Indian Institute of Technology, Delhi
Lecture: 01

Communication Between Nodes

(Refer Slide Time: 00:17)


Communication between Nodes
Epidemic Protocols
Gossip Based Protocols

Communication between Nodes
Epidemic Diffusion and Gossiping

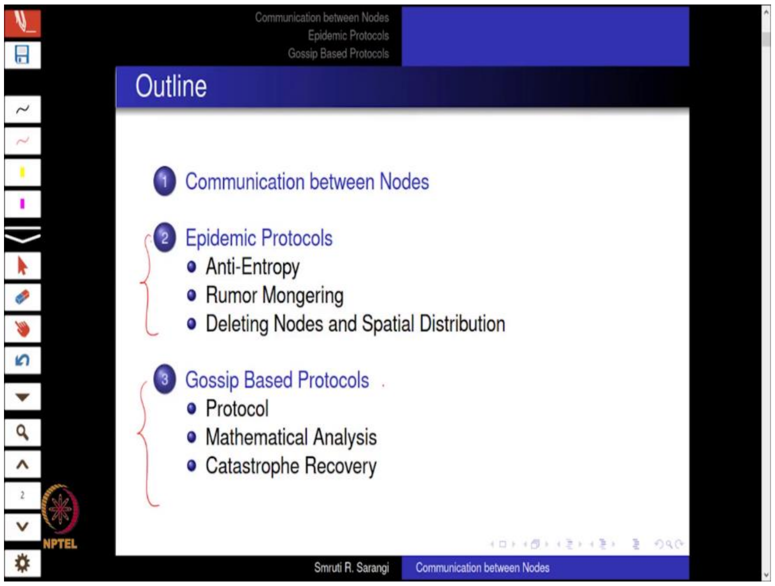
Smruti R. Sarangi

Department of Computer Science
Indian Institute of Technology
New Delhi, India

Smruti R. Sarangi   Communication between Nodes

In this lecture, we will discuss some of the basic mechanisms in which nodes in a distributed system communicate. So, these are very large distributed systems. So, in such distributed systems, often a node only knows the IDs of some of the nearby nodes, it does not clearly know which nodes comprise the entire network. So, this is not node. So, the communication between nodes in such cases is tricky, because given the fact that we do not know all the nodes in the network, we will not know if the message is reaching or not.

(Refer Slide Time: 00:59)

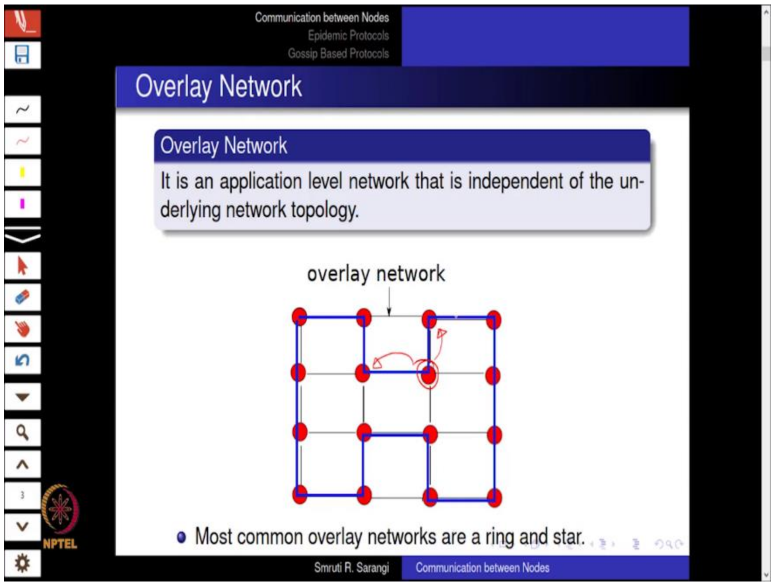


Outline

1. Communication between Nodes
2. Epidemic Protocols
   - Anti-Entropy
   - Rumor Mongering
   - Deleting Nodes and Spatial Distribution
3. Gossip Based Protocols
   - Protocol
   - Mathematical Analysis
   - Catastrophe Recovery

So, what we will discuss in this lecture set, in this lecture actually, in this video, are basically two kinds of protocols, Epidemic Protocols and Gossip Based Protocols, say Epidemic Protocols are clearly more popular in the literature, such as Anti-Entropy and Rumor Mongering, in comparison, Gossip Based Protocols are not that popular, but nevertheless they are also used. And we will see that they are heavily used in a lot of the distributed systems, particularly in modern systems where, we are talking of large web scale systems. So, these distributed systems are kind of distributed all over the internet.

(Refer Slide Time: 01:52)



Overlay Network

Overlay Network

It is an application level network that is independent of the underlying network topology.

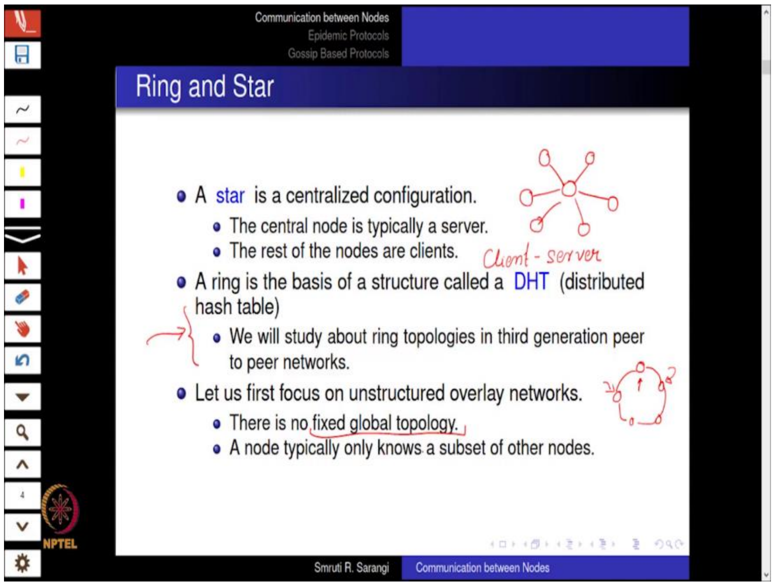
- Most common overlay networks are a ring and star.

So, the key idea of a distributed system is to form an overlay network. An overlay network is basically a virtual network, essentially a network over a network. So, regardless of how the physical organization of the network is? so, they could be wireless links, they could be Ethernet links, there are a wide variety of possibilities.

So, regardless of that, we create a virtual network. And that is known as the overlay. Where as you can see, this is a ring shaped overlay, where each node just knows the ID of its left neighbor and the ID of its right neighbors. Essentially, it is clockwise neighbor and its anti clockwise neighbor.

And we shall see that such overlays form the foundation of many of today's peer to peer networks. And most of our enterprise networks use such kind of overlays. And most of the common ones are clearly a ring and a star. So, this is more like a structured network.

(Refer Slide Time: 03:05)



Ring and Star

- A star is a centralized configuration.
  - The central node is typically a server.
  - The rest of the nodes are clients.
- A ring is the basis of a structure called a DHT (distributed hash table)
  - We will study about ring topologies in third generation peer to peer networks.
- Let us first focus on unstructured overlay networks.
  - There is no fixed global topology.
  - A node typically only knows a subset of other nodes.

So, the definition of a star is something that all of you know, but nevertheless, might not be a bad idea to define where we have one central node that can be like a server, or some sort of a base server. And then we have many other client nodes connected to it. So, a typical client server computing, where we have multiple clients and a single server would use such kind of a star shaped overlay.

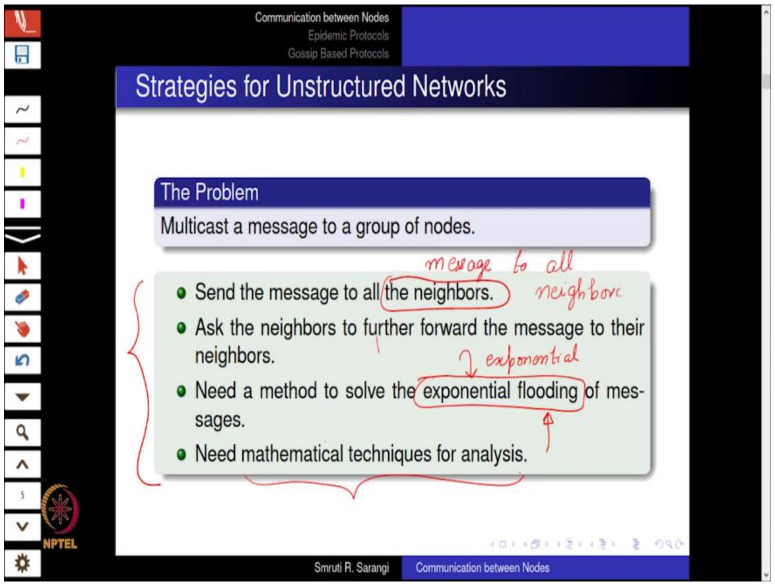
So, why do I call it an overlay? because this is not the real organization of the network. It is a virtual organization. So, given that it is a virtual organizational network, I am using that term overlay. A ring in comparison is way more popular. A ring is the basis of a structure called a distributed hash table or a DHT.

And they started coming in, in a big way third generation peer to peer networks. So, here what we do is that we can have a large system, over the internet, but that is, regardless of where they are, we just connect them as a virtual circle as a virtual drink. So, star shaped overlays and ring shaped overlays formed the basis of what we call structured networks.

But our entire discussion here has centered on centered around unstructured overlay networks where there is no fixed global topology. So, fix global topology is definitely not there are no typically nodes, a subset of other nodes, and a node is pretty much blind. Somebody can argue that in a ring shipped overlay also a node is pretty much blind in the sense this node only knows about its clockwise successor and anti clockwise successor.

But at least there is a structure in an unstructured overlay network there is no structure. so, there is no global structure you only have a small local structure and that too if there is one, sometimes you just know the list of a few nodes, that is it.

(Refer Slide Time: 05:28)



Communication between Nodes
Epidemic Protocols
Gossip Based Protocols

Strategies for Unstructured Networks

The Problem
Multicast a message to a group of nodes.

- Send the message to all the neighbors. message to all neighbors
- Ask the neighbors to further forward the message to their neighbors.
- Need a method to solve the exponential flooding of messages. exponential
- Need mathematical techniques for analysis.

Smruti R. Sarangi   Communication between Nodes

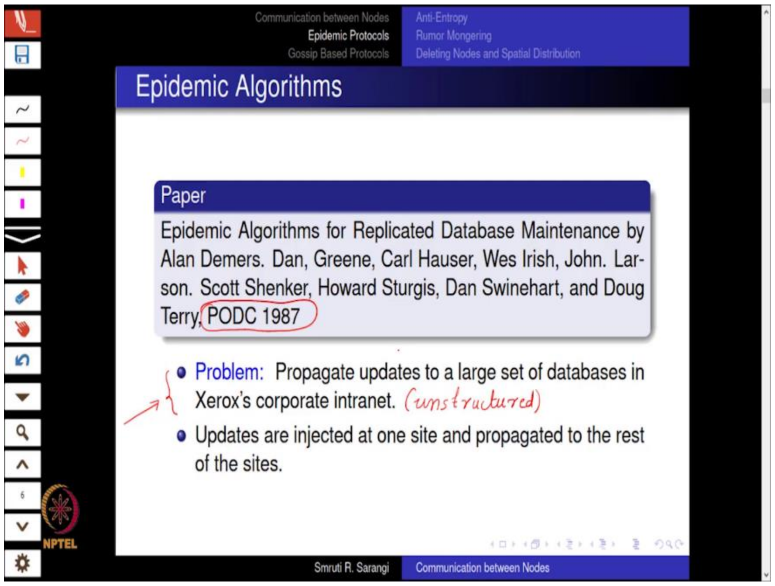
So, when we are talking of such unstructured networks, there is a problem, if I want to send a message or multicast a message to a group of nodes, there clearly is a problem and the problem is that I may not be able to reach the entire network, because I will not be aware of the state of the entire network.

And since I am not aware of the state that will to a large extent, cause a problem. So, this is not there with structured networks with a ring or a star, but we are mainly looking at unstructured networks here and there is an issue. So, what I can do is that I can send a message to all the neighbors. I can ask the neighbors to forward messages to their neighbors and so on.

So, this will kind of exponentially flood the network. But the negative aspect of this will create an exponential number of messages, which will become almost impossible to handle and there is no guarantee that the entire network of nodes will actually be reached. So, we need some way of kind of roping this in that pure exponential way is a bad way.

And furthermore, we need some mathematical techniques for analysis. For analyzing the system, for analyzing this setup, we need some mathematical techniques.

(Refer Slide Time: 07:12)



Epidemic Algorithms

Paper

Epidemic Algorithms for Replicated Database Maintenance by Alan Demers. Dan, Greene, Carl Hauser, Wes Irish, John. Larson. Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry, PODC 1987

- Problem: Propagate updates to a large set of databases in Xerox's corporate intranet. (unstructured)
- Updates are injected at one site and propagated to the rest of the sites.

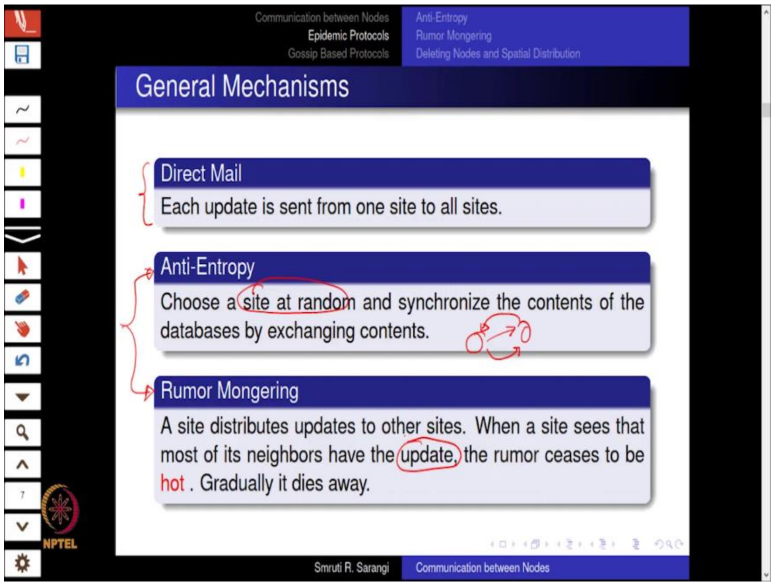
So, what I will discuss at the beginning, is this paper Epidemic Algorithms for replicated database maintenance. By as you can see a long list of authors published in PODC 1987. So, what was the basic problem that was being solved? The basic problems, was that there used to be a big company called Xerox.

So, we do not get to hear Xerox a lot in 2020. But Xerox at 1 point was a preeminent power in computing technology. So, for example, the graphical windows that you see, large part of that came from Xerox, the mouse came from Xerox. So, Xerox PARC Research Lab, used to be counted as maybe the world's best research lab in this area.

There was a competition with Xerox and IBM. So, Xerox had a large corporate intranet, which did not have a structure, it was unstructured. And they were maintaining databases. So, these databases that were being maintained, they needed to be kept in sync, which meant that updates had to be propagated, and how to propagate updates given that there were so many machines.

And those days, network communication was expensive. In terms of time, at least, this was a problem. So, updates are injected, typically at one site, and they have to be propagated to the rest of the sites. And given that one site did not have a global list, this became a problem.

(Refer Slide Time: 08:54)



Communication between Nodes | Epidemic Protocols | Gossip Based Protocols
Anti-Entropy | Rumor Mongering | Deleting Nodes and Spatial Distribution

General Mechanisms

Direct Mail
Each update is sent from one site to all sites.

Anti-Entropy
Choose a site at random and synchronize the contents of the databases by exchanging contents.

Rumor Mongering
A site distributes updates to other sites. When a site sees that most of its neighbors have the update, the rumor ceases to be hot . Gradually it dies away.

Smruti R. Sarangi | Communication between Nodes

So, there are three main approaches that we should look at. The three main approaches are Direct Mail, for update is sent from one site to all sites, the reason that direct mail could not be used was basically because number one, there were too many sites. And their exact list was not known. And furthermore, this is a sequential process.

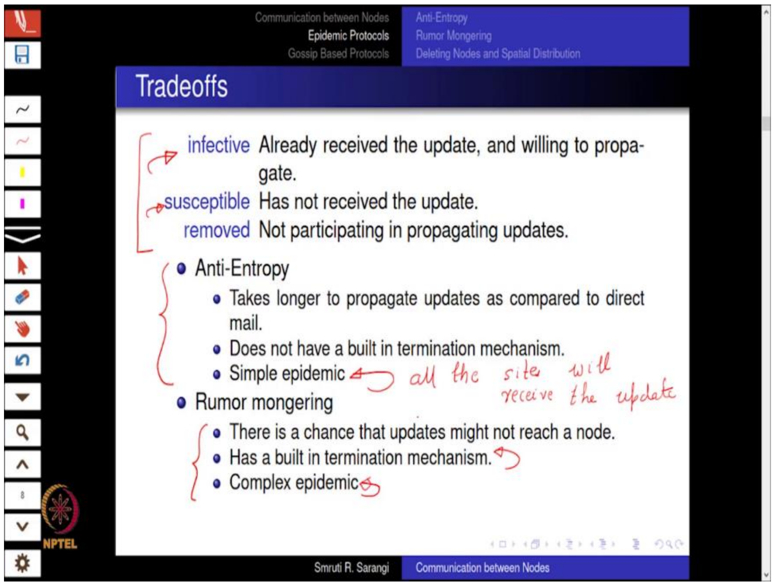
It is not a parallel process, it will take a lot of time to send messages and we are dealing with a slow internet, slow intranet situation over here. So, that is the reason the two broad approaches that were used are Anti-Entropy and Rumor Mongering. And these approaches later on were worked on by a generation of researchers.

And today there are very sophisticated Anti-Entropy and Rumor Mongering methods, which are actually used by many of the companies that effect our lives such as Amazon, Facebook, LinkedIn, and so on, of course, not the same way that Xerox used, but in a modified version that is suitable to those companies, suitable to the systems of those companies and improved.

So, Anti-Entropy says that we choose a site at random. And we synchronize the contents of the database by exchanging contents, which means that I randomly choose a site, I give it my updates, and I take the updates of that site and update my own database. So, both of us become both of us know, whatever we knew earlier, in the sense that let us say whatever is new, the other side knows and whatever the other side knows that I do not know that is transferred to me. So, our knowledge levels become equal. This is Anti-Entropy.

So, Anti-Entropy, by definition does not stop because we keep on finding sites at random and we keep on exchanging information with them. The other approach is Rumor Mongering, where a site distributes updates to other sites. When a site sees that most of his neighbors have the update, it ceases to transmit updates with that frequency reduces its frequency of prospecting updates. So, we say that this is like spreading a rumor. So, the rumor in this case is the update. So, that rumor ceases to be hot, and gradually it dies away, it fades away.

(Refer Slide Time: 11:39)



Tradeoffs

infective	Already received the update, and willing to propagate.
susceptible	Has not received the update.
removed	Not participating in propagating updates.

- Anti-Entropy
  - Takes longer to propagate updates as compared to direct mail.
  - Does not have a built in termination mechanism.
  - Simple epidemic ← all the site will receive the update
- Rumor mongering
  - There is a chance that updates might not reach a node.
  - Has a built in termination mechanism.
  - Complex epidemic

So, we will do a little bit of a mathematical analysis over here. So, we will create three kinds of sites, infective, susceptible and removed. An infective site is something that has already received the update and willing to propagate. So, incidentally, these terms have been taken from the literature of epidemiology, which sees how epidemics propagate?

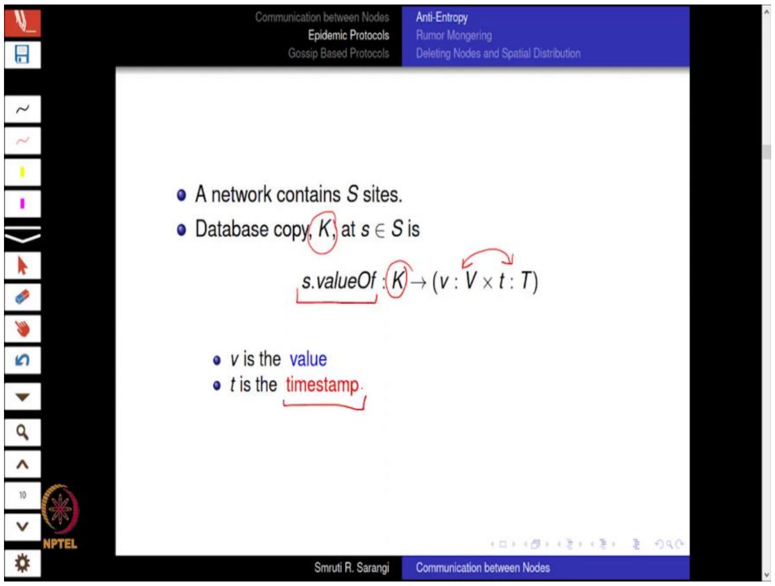
And given that I am recording this video in the middle of the corona virus pandemic, these terms will appear to be extremely relevant. So, an infective is essentially a person in this case a site that is willing to propagate susceptibility somebody who has not a site basically, that has not received the update.

So, it can be infected later and removed is that you are not participating in propagating updates. So, in a sense, you have disconnected yourself from the network as far as updates are concerned, so that is the terms in anti entropy, it takes longer to propagate updates as compared to direct mail.

That is true, it does not have a built in termination mechanism, but it is what is called a simple epidemic, in the sense that it is easy to propagate information, and that to very easily it is possible to propagate. And it is nice and simple is a simple epidemic. In comparison, rumor mongering is a complex epidemic, because it has a built in termination mechanism.

But unlike anti entropy, there is a problem. So, anti entropy guarantees that look ultimately all the sites will have received the update. But in rumor mongering there is a chance that updates (())(14:02) reach a node which basically means that there is a chance there is a possibility or probability that the updates might not reach a certain node. A node in this case is a site.

(Refer Slide Time: 14:16)


Communication between Nodes
Epidemic Protocols
Gossip Based Protocols
Anti-Entropy
Rumor Mongering
Deleting Nodes and Spatial Distribution
Smruti R. Sarangi
Communication between Nodes
NPTEL

- A network contains S sites.
- Database copy, K, at $s \in S$ is

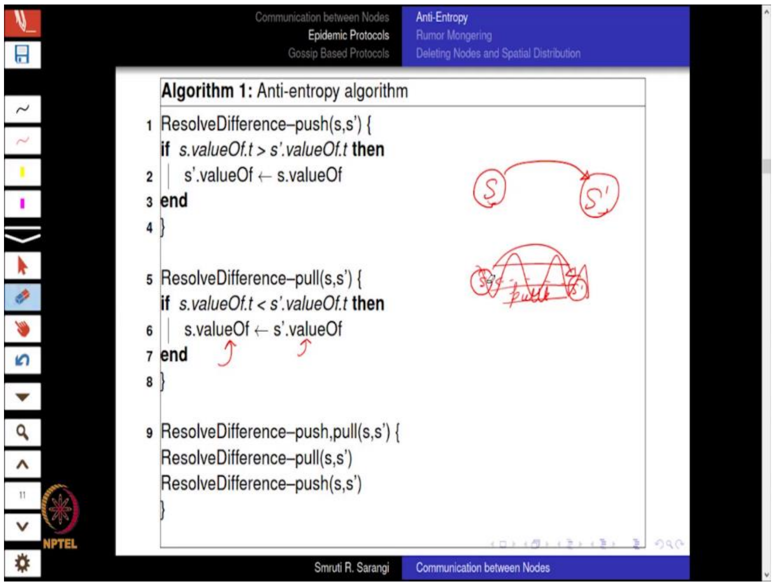
$$s.valueOf : K \rightarrow (v : V \times t : T)$$

- v is the value
- t is the timestamp.

So, let us start with Anti-Entropy. So, let us say that a network contains S sites and the database copy K at a given site smallest. Its value is essentially a tuple. The tuple K has two parts to it. The first is the actual value of the update and the T is the timestamp. So, basically any newer update will have a higher timestamp. So, this is how we differentiate between old update a new update, it was necessary to add this timestamp in the network.

Because it is possible that messages that circulate older updates might still be alive in the network. And this is how we will distinguish between old updates and new updates. And clearly old updates should be discarded and the new updates are the ones that should be applied and this will be decided on the basis of the timestamp.

(Refer Slide Time: 15:30)

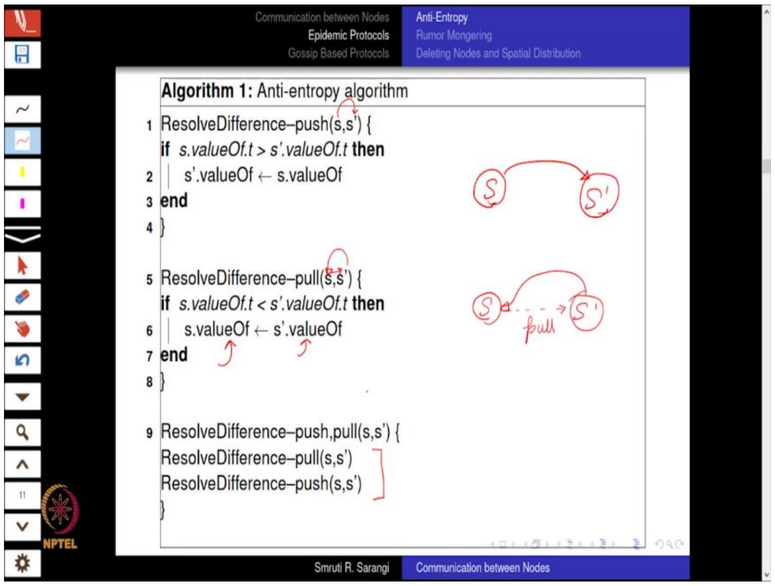


Algorithm 1: Anti-entropy algorithm

```
ResolveDifference–push(s,s') {
if s.valueOf.t > s'.valueOf.t then
    s'.valueOf ← s.valueOf
end
}

ResolveDifference–pull(s,s') {
if s.valueOf.t < s'.valueOf.t then
    s.valueOf ← s'.valueOf
end
}

ResolveDifference–push,pull(s,s') {
ResolveDifference–pull(s,s')
ResolveDifference–push(s,s')
}
```



Algorithm 1: Anti-entropy algorithm

```
ResolveDifference–push(s,s') {
if s.valueOf.t > s'.valueOf.t then
    s'.valueOf ← s.valueOf
end
}

ResolveDifference–pull(s,s') {
if s.valueOf.t < s'.valueOf.t then
    s.valueOf ← s'.valueOf
end
}

ResolveDifference–push,pull(s,s') {
ResolveDifference–pull(s,s')
ResolveDifference–push(s,s')
}
```

So, in the anti entropy algorithm per se is easy. So, we have two kinds of anti entropy push and pull. So, there are two sides S and S’, what push does is it pushes the updates of S to S’. So, basically sees that if S has a higher timestamp, then it sets S’ value to s the difference between push and pull is that in a pool S’ essentially sends a pull message and it asks as for the most recent copy of data that it has.

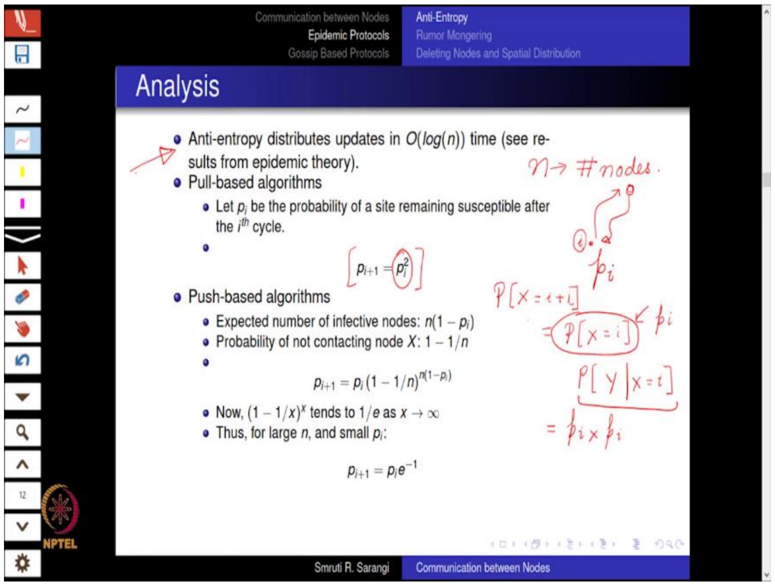
So, the pull basically again does the same thing that is S the lower timestamp than S’, then the s s value is set equal to S’ value. So, it is important to look at this once again. So, in push what we are doing is S is pushing to a S’.

In pull what we are doing is that S is essentially asking S' look do you have any data, that might be of potential use to me. So, if S has a lower timestamp, then what it will do is it will kind of pull the data from S' as s will set the value of its data to S' value. Of course the S' data has recent more recent timestamp.

So, let us temporarily it is this figure So, pull what happens is that we have two sites S and S'. So, s sensate a pull message saying do you have any updates for me?

And if it does, then S' sensate an update? S compares a timestamp if it has a smaller timestamp well then it takes up a S' update. So, what is the semantics of push and pull when push S is pushing its updates to S' and in pull s s is asking a S' look do you have a new update? Anyway S' has a new update it sends it to S and clearly if from s s point of view if it is nill s applies it. A push pull is a combination of both the scheme's both push as well as pull. And so that is the reason I am not describing it separately.

(Refer Slide Time: 18:28)



Communication between Nodes | Anti-Entropy
Epidemic Protocols | Rumor Mongering
Gossip Based Protocols | Deleting Nodes and Spatial Distribution

Analysis

- Anti-entropy distributes updates in $O(log(n))$ time (see results from epidemic theory).
- Pull-based algorithms
  - Let p_i be the probability of a site remaining susceptible after the i^{th} cycle.
  - $$\left[p_{i+1} = p_i^2 \right]$$
- Push-based algorithms
  - Expected number of infective nodes: $n(1 - p_i)$
  - Probability of not contacting node X: $1 - 1/n$
  - $$p_{i+1} = p_i (1 - 1/n)^{n(1-p_i)}$$
  - Now, $(1 - 1/x)^x$ tends to $1/e$ as $x \to \infty$
  - Thus, for large n, and small p_i:

$$p_{i+1} = p_i e^{-1}$$

$n \rightarrow$ #nodes.

p_i

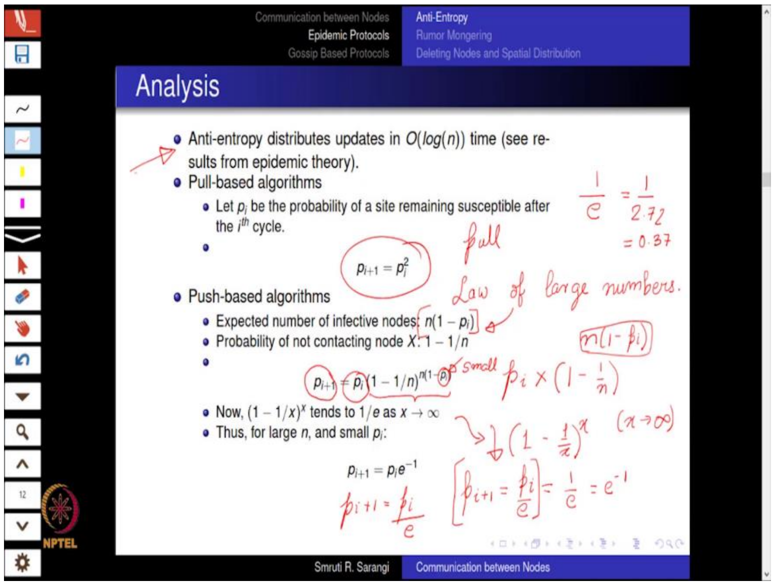
$P[X = i+1]$

$= \boxed{P[X = i]} \leftarrow p_i$

$\underbrace{P[Y \mid X = i]}$

$= p_i \times p_i$

Smruti R. Sarangi | Communication between Nodes


Communication between Nodes | Anti-Entropy
Epidemic Protocols | Rumor Mongering
Gossip Based Protocols | Deleting Nodes and Spatial Distribution

Analysis

- Anti-entropy distributes updates in $O(log(n))$ time (see results from epidemic theory).
- Pull-based algorithms
  - Let p_i be the probability of a site remaining susceptible after the i^{th} cycle.
  - $$\boxed{p_{i+1} = p_i^2}$$
- Push-based algorithms
  - Expected number of infective nodes: $n(1 - p_i)$
  - Probability of not contacting node X: $1 - 1/n$
  - $$p_{i+1} = p_i \underbrace{(1 - 1/n)^{n(1-p_i)}}$$
  - Now, $(1 - 1/x)^x$ tends to $1/e$ as $x \to \infty$
  - Thus, for large n, and small p_i:

$$p_{i+1} = p_i e^{-1}$$

$\frac{1}{e} = \frac{1}{2.72} = 0.37$

pull

Law of large numbers.

$\boxed{n(1-p_i)}$

small

$p_i \times (1 - \frac{1}{n})$

$\left(1 - \frac{1}{x}\right)^x \quad (x \to \infty)$

$p_{i+1} = \frac{p_i}{e}$

$\left[p_{i+1} = \frac{p_i}{e}\right] = \frac{1}{e} = e^{-1}$

Smruti R. Sarangi | Communication between Nodes

So, let us now do a little bit of math and understand what exactly is going on here. So, what we are out to prove is that anti entropy distributes updates in order of login time. This is a result from epidemic theory. Try to apply this to a corona virus scenario. In this case, n is the total number of nodes.

So, in a pool based algorithm, let pi be the probability of a site remaining susceptible after the ith cycle. So, for a given site, let us say after the ith cycle, the probability it is still susceptible is pi. What is the probability that it is susceptible even after the i + 1 ith cycle. Well, what would happen? We are dividing this into rounds, every round.

The ith node over here will contact some other node. Because it is a pull based system. And if the other known as a recent update, it will get it. If that is the idea, then let us see. So, this is essentially the conditional probability over here is that what is the probability of the other site to actually contain an update, or rather not contain an update?

So, if it does not contain an update, and since we do not remove nodes and anti entropy, it will still be susceptible? So, how do I write the probability? Well, the probability, let us say that that the event that I am susceptible, let us say that event is P [x = i + 1], which means ith cycle I am susceptible, is the probability that I am susceptible in P [x = i] cycle, multiplied with a probability that let us see, I contact a node which is given that I was susceptible in P [y | x = i] cycle, I contact a node lead that event y and that node is susceptible. That node is not infected. So, then the first probability is clearly pi.

And the second is that I contact a subset of nodes, which are not infected. So, the subset of nodes that are not infected, that is again equal to pi. So, the net probabilities pi into pi. So, what am I trying to say I am trying to say that look, this quantity is pi by definition. This quantity, the fact that I contact some other known and that turns out to be susceptible, again, arises from the definition of probability that is the way probability is defined that it is the size of the interested points divided by the size of the sample space.

So, in this case, the number of interested points, which are the ones that are not infected, still susceptible, divided by the sample space is pi. So, that is where the second pi comes from. If I multiply them, I get Pi^2. So, if you are not able to understand this explanation, I would suggest that hold on, let me just try to see if I can clean off parts of the page.

So, as I mentioned if you are not able to understand what I just said, I would suggest that to rather strongly that you look at texts in probability and that will kind of clear up most of your doubts. So, that will clear up and that will tell you why? We were able to get pi squared over here, which is just a product of two things the product the fact that it is susceptible, and ith cycle multiplied by the probability, that if it is susceptible, ith cycle, what is the probability that it will be susceptible and i + 1, ith cycle, both are equal to pi.

So, that is where we get pi^2. This will happen in a pull based algorithm not in a push based. So, the push based algorithm again, this comes from the law of large numbers, this is the right time to go to Wikipedia and study the law of large numbers. So, the expected number infective nodes, the nodes that are not susceptible, is n (1 – pi).

That should be, the probability a node is effective is 1 - pi and the total expected number is n times this, this follows from the law of large numbers the weak law in particular, the probability that I do not contact a node is 1 - $\frac{1}{n}$. So, what is the probability that a given node is susceptible and i + 1 ith cycle?

Well, again, it is the probability that the node is susceptible in the ith cycle, so, the probability that this node is susceptible in ith cycle multiplied with the probability that no other node contacts it in ith round, which is infective. So, no infective node contacts it in the ith round, that is important. So, how many such nodes are there? Well, so the point is that there are n (1 – pi) such nodes, none of these nodes contacted me.

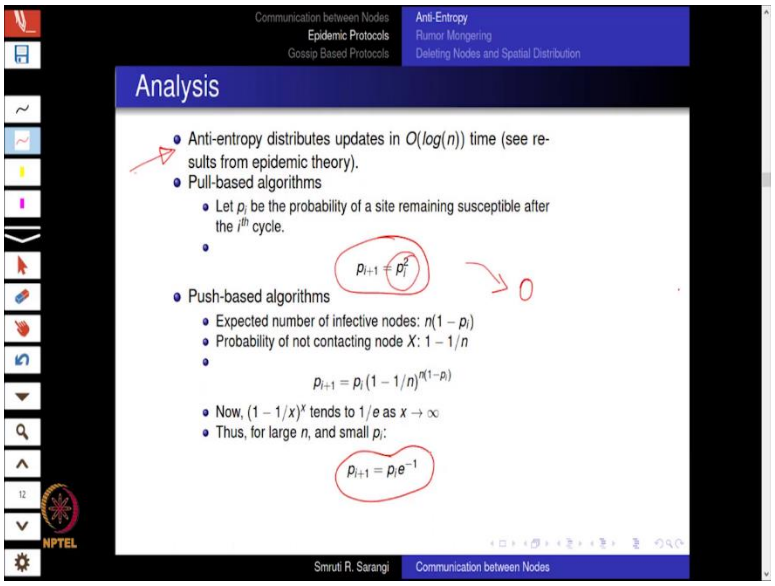
So, for each node, the probability of not contacting me is $(1-\frac{1}{n})^{n\,(1-pi)}$, which is essentially this quantity is the probability that I remain susceptible and ith round, given the fact that at the beginning of the ith round, never susceptible. So, we will use a well-known result in calculus, which is $(1-\frac{1}{x})^{\,x}$ as x → ∞ = $\frac{1}{e}$, or e^{-1}.

So, this again is a result from calculus that we shall use, and this will tell us that (pi + 1 = pi). well, pi comes from there. So, (1 - $\frac{1}{n})^{n}$. so, of course, here an assumption that I am making here is that pi is rather small. So, this assumption that the probability that a node is still susceptible, let us say that this probability is kind of small.

So, we can approximate this to e^{-1} which is $\frac{pi}{e}$. So, if I go with a small, since even written that is a large n and small pi, I can approximately have this assumption that the probability of susceptibility every round falls by a factor of $\frac{1}{e}$, $\frac{1}{e}$ you would have encountered in many-many others scenarios it is around 0.37.

So, roughly let us say that every round the probability of susceptibility falls down with 37%. So, after a few rounds itself, it will become small very small, and then it will continue to decrease, but mind you this assumption is only valid for a small period.

(Refer Slide Time: 27:52)

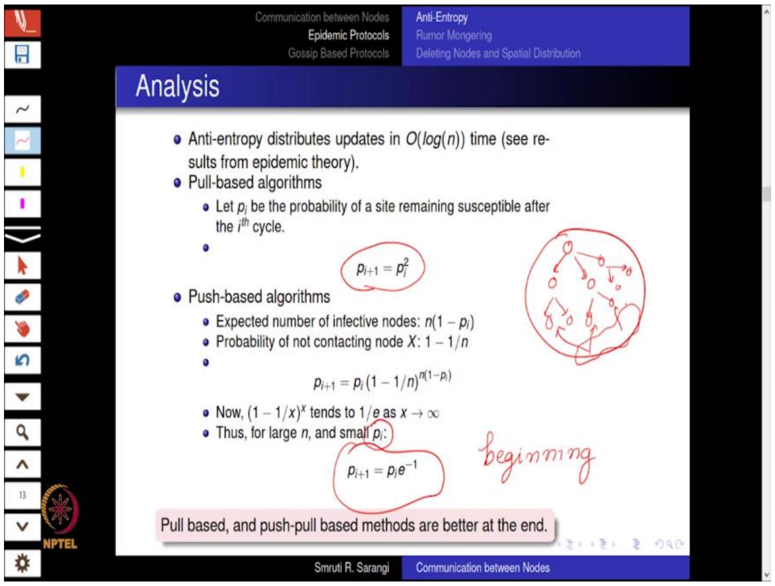


Analysis

- Anti-entropy distributes updates in $O(log(n))$ time (see results from epidemic theory).
- Pull-based algorithms
  - Let p_i be the probability of a site remaining susceptible after the i^{th} cycle.
  - $$p_{i+1} = p_i^2 \rightarrow 0$$
- Push-based algorithms
  - Expected number of infective nodes: $n(1 - p_i)$
  - Probability of not contacting node X: $1 - 1/n$
  - $$p_{i+1} = p_i(1 - 1/n)^{n(1-p_i)}$$
  - Now, $(1 - 1/x)^x$ tends to $1/e$ as $x \rightarrow \infty$
  - Thus, for large n, and small p_i:

$$p_{i+1} = p_i e^{-1}$$

12
NPTEL
Smruti R. Sarangi
Communication between Nodes


Communication between Nodes
Epidemic Protocols
Gossip Based Protocols
Anti-Entropy
Rumor Mongering
Deleting Nodes and Spatial Distribution

Analysis

- Anti-entropy distributes updates in $O(log(n))$ time (see results from epidemic theory).
- Pull-based algorithms
  - Let p_i be the probability of a site remaining susceptible after the i^{th} cycle.
  - $$p_{i+1} = p_i^2$$
- Push-based algorithms
  - Expected number of infective nodes: $n(1 - p_i)$
  - Probability of not contacting node X: $1 - 1/n$
  - $$p_{i+1} = p_i(1 - 1/n)^{n(1-p_i)}$$
  - Now, $(1 - 1/x)^x$ tends to $1/e$ as $x \rightarrow \infty$
  - Thus, for large n, and small p_i:

$$p_{i+1} = p_i e^{-1}$$

beginning

Pull based, and push-pull based methods are better at the end.

So, let us look at these two expressions. So, if pi is small enough, then basically this expression reduces a squared because given that these are probabilities, these are actually fractions this essentially get squared and then to the power 4 times, 8 times, so on and so forth. Whereas, this reduces by a constant, so, pull will reduce the probability of susceptibility which should ideally become 0 at the end.

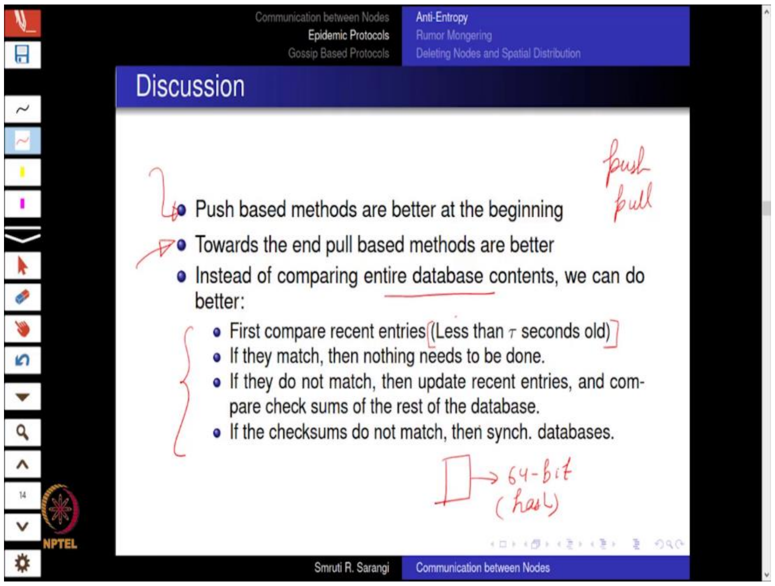
So, this should tend to 0 as a square and this is why a constant. So, we need to now understand which one is better. So, the point is that it all depends on whether it is the beginning or whether it is the end. So, let us say this is a big network. At the beginning, let us see if we use a pull based method, then nobody is going to get anything. The reason that nobody is going to get anything is because there are very few susceptible sites.

So, there will be no advantage will nobody is nearly going to contact some other few sites that are still infected. So, at the beginning, push based methods are clearly better because at least they are reducing it by a constant factor. Of course, we have assumed small pi but even if let us say you were to do a simulation, you will still see it reducing quite significantly, quite quickly as compared to pull based methods.

So, at the beginning, definitely a push based method is preferred. And what the push base method actually does is that it propagates the updates to as many nodes as you can, as many as it can. Gradually, as the updates have disseminated, there will still be a small minority of nodes that have not gotten the update.

Well, then they can revert to a pull based method, where they will contact nodes that have reached the bottom to update or infective and then they can fetch the update from there. That is why we say that pull based methods are better at the end.

(Refer Slide Time: 30:08)



Discussion

- Push based methods are better at the beginning
- Towards the end pull based methods are better
- Instead of comparing entire database contents, we can do better:
  - First compare recent entries (Less than τ seconds old)
  - If they match, then nothing needs to be done.
  - If they do not match, then update recent entries, and compare check sums of the rest of the database.
  - If the checksums do not match, then synch. databases.

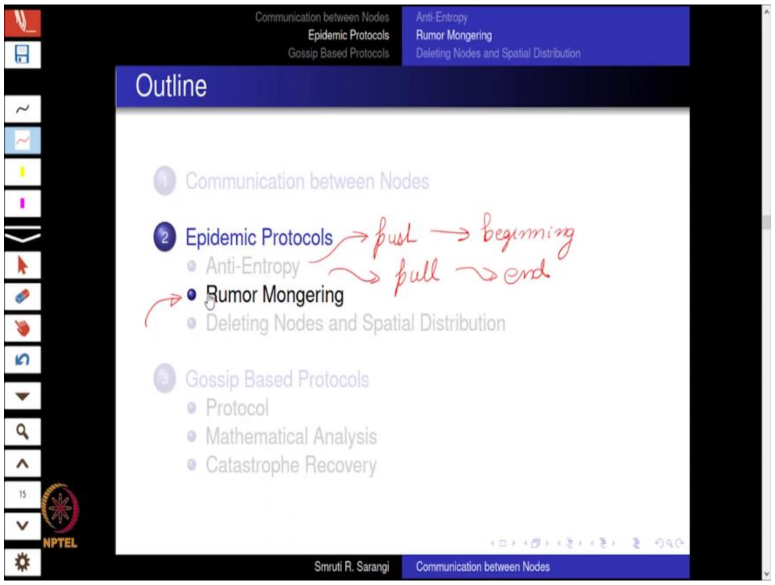
So, what was the thing? from math, we can say that look at the beginning push based is better, because, you are just coming into the network with an update. So, contact as many people and spread the update. How it, twits the end, pull based methods are better. Because even if let us say we are trying to do push, I will still not be able to contact the ones with high probability, of course that are not received the update.

But in pull base methods, the nodes that are not received, the update will continue fetching them. And as they get into a minority the probability of they actually hitting a node, that has the update kind of increases. So, that is why any method has pushed at the beginning pull at the end. So, this is anti-entropy for you can be optimized. Instead of comparing the entire database contents, well compare the timestamps of the recent entries.

If the timestamps match, well, nothing much needs to be done. Because it will tell you that earlier entries are there. And if they do not match, well then update the recent entries and compare what are called check sums. Check sums are, essentially think of them as advanced versions of parity bits, where let us say if you have a 64-bit check sum, it kind of uniquely identifies the contents of the entire database.

So, the entire contents of the entire database, is kind of compressed to a 64-bit number. And that uniquely identifies (()) (31:55) hash. It is a hash the checksums do not match which means the hashes do not match means that there will be one small little difference somewhere. So, there is a need to transfer the entire databases and sync it otherwise per se, there is no need, we can compare hashes for older entries and for recent entries, very recent entries are there we can compare the entries itself and entries themselves.

(Refer Slide Time: 32:24)


Communication between Nodes
Epidemic Protocols
Gossip Based Protocols

Anti-Entropy
Rumor Mongering
Deleting Nodes and Spatial Distribution

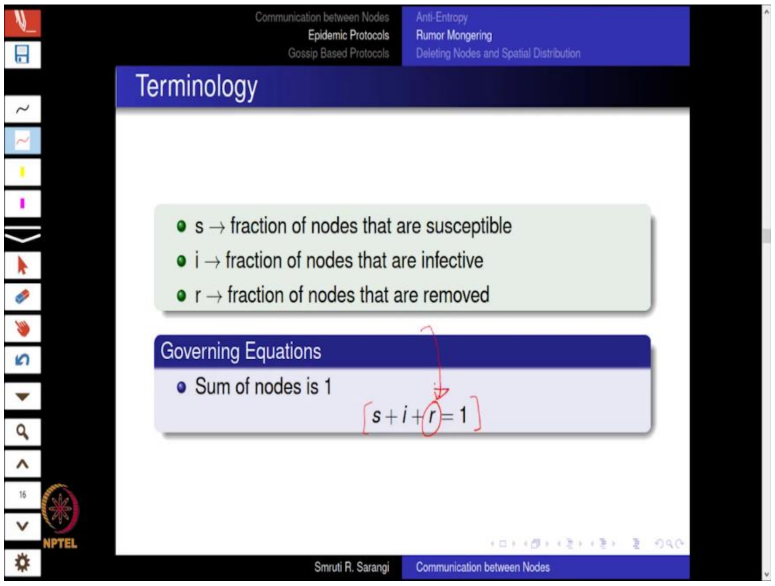
Outline

1. Communication between Nodes
2. Epidemic Protocols $\to$ push $\to$ beginning
   - Anti-Entropy $\to$ pull $\to$ end
   - Rumor Mongering
   - Deleting Nodes and Spatial Distribution
3. Gossip Based Protocols
   - Protocol
   - Mathematical Analysis
   - Catastrophe Recovery

Smruti R. Sarangi — Communication between Nodes

So, we have looked at anti entropy, we have looked at the push based mechanism we have looked at the pull based mechanism, we have said that look, this is better at the beginning. And pull is better at the end. Now, we look at rumor mongering which is a second kind of epidemic protocol. So, why epidemic protocol because these protocols have come out of epidemic theory, something that you see in excess in these COVID pandemic days.

(Refer Slide Time: 33:05)


Communication between Nodes
Epidemic Protocols
Gossip Based Protocols

Anti-Entropy
Rumor Mongering
Deleting Nodes and Spatial Distribution

Terminology

- s → fraction of nodes that are susceptible
- i → fraction of nodes that are infective
- r → fraction of nodes that are removed

Governing Equations

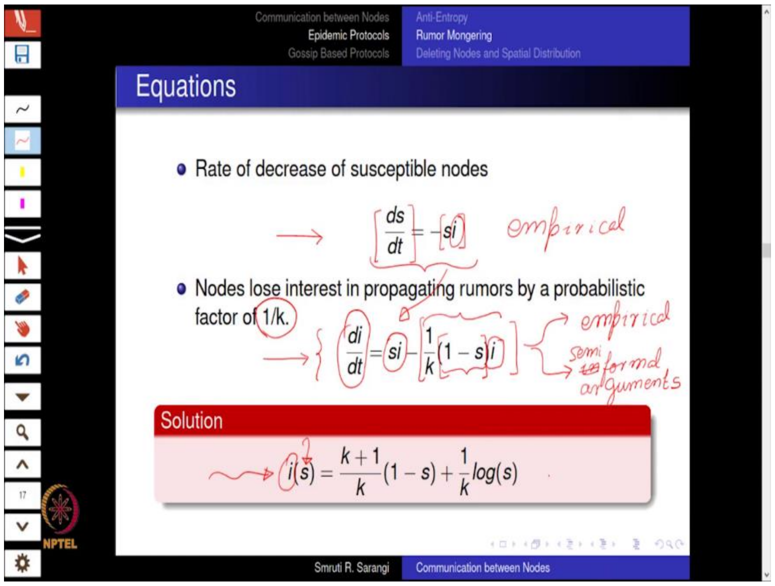
- Sum of nodes is 1

$$\left[s + i + \textcircled{r} = 1 \right]$$

Smruti R. Sarangi | Communication between Nodes

So, here also will have the same idea of susceptible, infective and removed. In this case, it is not the number of nodes but rather the fraction. So, that is why we have s + i + r = 1. So, these are just the fractions. So, recall that this remove category was not there in anti-entropy, but the remote category is there in the rumor mongering, s + i + r = 1.

(Refer Slide Time: 33:39)


Communication between Nodes
Epidemic Protocols
Gossip Based Protocols

Anti-Entropy
Rumor Mongering
Deleting Nodes and Spatial Distribution

Equations

- Rate of decrease of susceptible nodes

$$\longrightarrow \underbrace{\boxed{\frac{ds}{dt}} = -\boxed{si}} \quad \text{empirical}$$

- Nodes lose interest in propagating rumors by a probabilistic factor of $\textcircled{1/k}$.

$$\longrightarrow \left\{ \boxed{\frac{di}{dt}} = \textcircled{si} - \frac{1}{k}\left[(1-s)i\right] \right\} \begin{cases} \text{empirical} \\ \text{semi ~~...~~ formal arguments} \end{cases}$$

Solution

$$\rightsquigarrow i(s) = \frac{k+1}{k}(1-s) + \frac{1}{k}\log(s)$$

Smruti R. Sarangi | Communication between Nodes

So, we will use some calculus here to derive the results. So, let us look at the rate of decrease of susceptible nodes. So, this is $\frac{ds}{dt}$. So, $\frac{ds}{dt}$ will have a (-) sign because, well $\frac{ds}{dt}$ is pretty much the rate of increase, but the negative sign makes it the rate of decrease.

So, this is clearly proportional to two quantities. It is proportional to the fraction of susceptible nodes because clearly if there are more susceptible nodes then more nodes will get contacted, more nodes will get infected and more nodes will enter the infective category. It is also proportional to the number of infective nodes because more or the number of infective nodes.

Hired is the proportion of the number of infective nodes proportionately hired, is the proportion is the rate of decrease of susceptible nodes**.** So, well, these equations do hold, let us say for a steady state range. Fare let us say, the number of infective nodes is not much it is clearly not saturating. Similarly, the number of saturating nodes the number of susceptible nodes is also not extremely large, fair, this may not hold or it is not extremely small.

So, there are clearly ranges where this equation holds exactly and ranges where this equation does not hold all that well. But let us say that these are all empirical laws in the sense they have been observed to hold at least most of the time. And they also can be derived from the simple informal explanation that I gave that the rate of decrease of the susceptible nodes is in a sense, proportional to both the quantities and both the quantities are thus we consider a product of these quantities.

Furthermore, lead nodes lose interest in propagating rumors by a probabilistic factor of $\frac{1}{k}$, which means that nodes will gradually lose interest in propagating the infection in this case, it is a rumors with the rate $\frac{1}{k}$. So, this equation is derived like this the $\frac{di}{dt}$, which is again the rate of increase of infective nodes. Well, this for reasons similar to the previous reason is also proportional to si, which means proportional to the product of the fraction of both susceptible as well as infective nodes.

And the reason is similar to the reason that we gave over here however, there is a damping factor. So, let us understand this damping factor. So, this damping factor is clearly proportional to the number of infective nodes, which should be the case because higher is the number of infective nodes, higher should be the damping, it is also proportional to the number of nodes that are not susceptible.

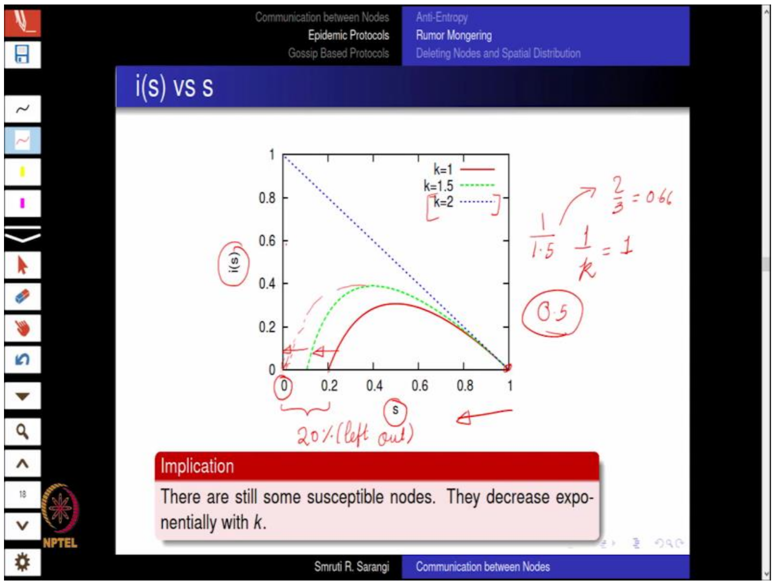
So, if we go back to this equation over here, then the nodes that are not susceptible are either infective or removed. So, basically it is proportional to (1 – s). So, (1 – s) is basically the non-susceptible part of the network and as that increases that does put in a damping effect and here is the factor $\frac{1}{k}$ that we talked about, and the factor k does determine how quickly the rumor dies or how quickly the rumor feeds away. So, as I have said the equations have been derived a combination of two things the first is empirical observations and they do tend to hold for a large range of i and s, they do this approximate relationship does hold and also from the informal arguments that I just provided up till now.

Well, informal is a bad word, let me call it the semi-formal arguments that have been provided to you. so, they basically look at the interaction of these two factors. And so, the interaction of these two factors pretty much determines how the numerator is going to propagate. So, if I were to solve both of these equations, which would be this equation over here, and this equation over here, if I were to solve them, then I can always find the rate of infective nodes, the fraction of infective nodes as a function of s.

So, this turns out that it is a sum of a linear function and a log function with the damping factor playing a very key role as you can see over here. So, what are we getting to see over here? what we are getting to see is that T we have eliminated and we basically have i as a function of s. And of course, we can see the damping factors play a key role. So this, the derivation of this is clearly given in the paper, and you can take a look at it. This is just a brief summary of what I am showing.

And given that we are in the middle of a pandemic. now, I would invite all of you to take some maybe Coronavirus data and try to fit it with whatever you can whatever you think is the damping factor. So, the damping factor is can be derived from what they are quoting as the odd number, which is the number of people that one person infects during this is his or her Coronavirus affliction. So, the damping factor can be derived from that. So, for different damping factors, you can fit these curves and see how the infection will evolve over time.

(Refer Slide Time: 40:49)



i(s) vs s

Implication

There are still some susceptible nodes. They decrease exponentially with k.

So, if I were to plot it as a fraction of s, so, initially of course, all the notes are susceptible. So, we will start from here and we will walk backwards this way. So, if we see k = 1. so if k = 1 what we get to see is that we see a curve like this. So, basically, the multiplicative damping factor is actually $\frac{1}{k}$. So, $\frac{1}{k}$ =1.

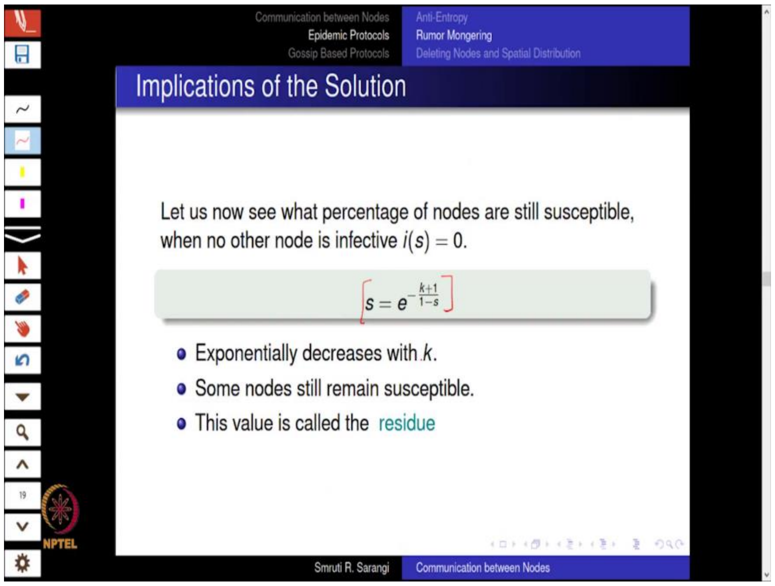
So, what we are seeing is that the data of infective nodes will increase, increase, increase and then abruptly fall to 0 not abruptly, but let us say in this range, it will fall to 0. So, ultimately, we will have Pheno number of infected nodes is 0, which means no more infections can actually happen, no more nodes will actually get the updates, we will see that around 20% of the nodes are effectively left out.

So, they are still susceptible or they are left out. So, this clearly depends on the damping factor. So, if I were to make k = 1.5, which will make that effective product $\frac{1}{1.5}$. So, we see a shift of this curve we see a shift of this curve towards the left and gradually as a damping factor k, I should actually call the damping factor is $\frac{1}{k}$.

So, gradually as that reduces or as k increases, we will reach this point and the moment that the curve actually reaches this point, it will basically tell us that look at this point, no node will be left. Uninfected in a sense all the susceptible nodes will become infected. So, there will be no node that is not susceptible, which means that it is infected. And of course I show the figure curve here where k = 2 and k = 2 which means $\frac{1}{k}$ = 0.5.

That is clearly too much in the sense we can see that all the nodes at the end will remain infective. And the number of susceptible nodes will actually fall down to 0. So, this is the key point over here, that at the end over here, all the nodes still remain infective and no node is susceptible and it is clearly not removed from the network.

(Refer Slide Time: 43:56)


Communication between Nodes
Epidemic Protocols
Gossip Based Protocols
Anti-Entropy
Rumor Mongering
Deleting Nodes and Spatial Distribution

Implications of the Solution

Let us now see what percentage of nodes are still susceptible, when no other node is infective $i(s) = 0$.

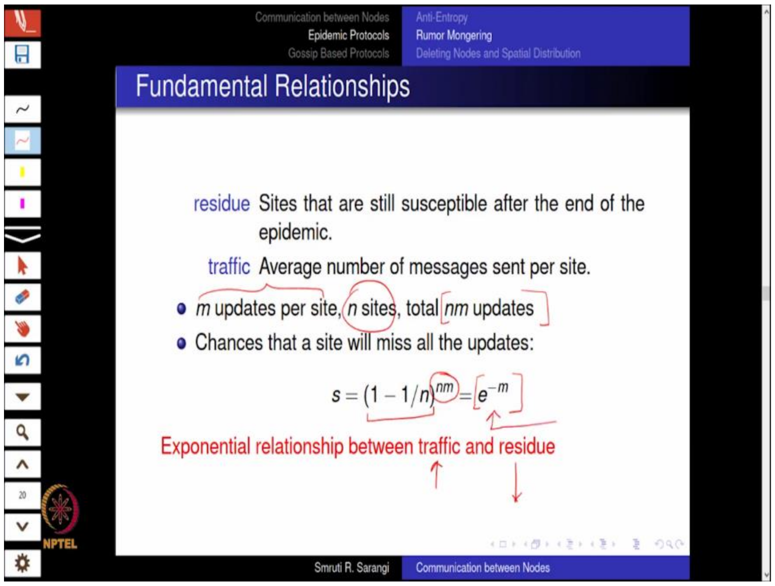
$$s = e^{-\frac{k+1}{1-s}}$$

- Exponentially decreases with k.
- Some nodes still remain susceptible.
- This value is called the residue

Smruti R. Sarangi
Communication between Nodes

So, now, let us see what percentage of nodes are still susceptible when no other node is infected that is i(s) = 0. So, which is essentially the nice point where we try to kind of estimate this residue over here which is this much. So, this can easily be derived to be this s = $e^{\frac{-k+1}{1-s}}$. So, we can see that it exponentially decreases with k and this value is called the residue and some nodes will still remain susceptible.

(Refer Slide Time: 44:32)


Communication between Nodes
Epidemic Protocols
Gossip Based Protocols
Anti-Entropy
Rumor Mongering
Deleting Nodes and Spatial Distribution
Smruti R. Sarangi
Communication between Nodes

Fundamental Relationships

residue Sites that are still susceptible after the end of the epidemic.

traffic Average number of messages sent per site.

- m updates per site, n sites, total nm updates
- Chances that a site will miss all the updates:

$$s = (1 - 1/n)^{nm} = e^{-m}$$

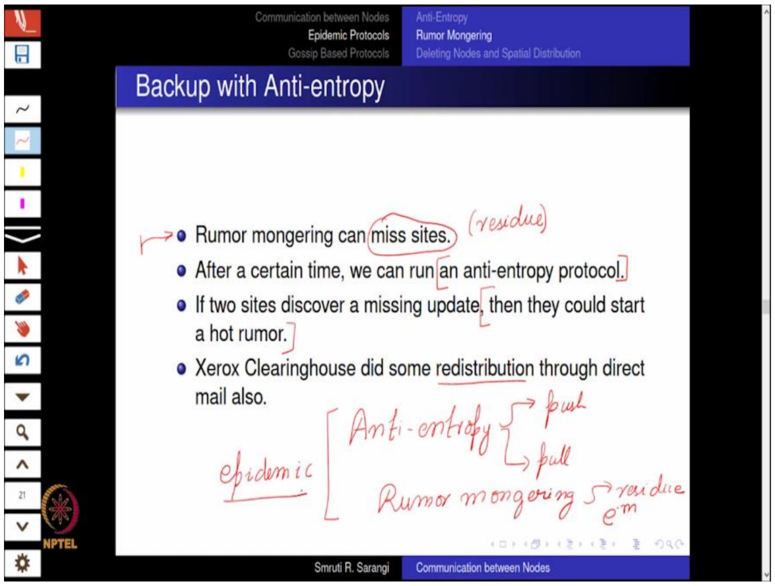
Exponential relationship between traffic and residue

So, let us now look at a few more fundamental relationships. So, the residue is the sites that are still susceptible after the end of the epidemic and the traffic is the average number of messages that we sent per site. So, let us say we send m updates per site, and there are a total of n sites. So, we will have a total of nm updates.

The chances that a site will miss all the updates and still remain susceptible is (1 - $\frac{1}{n}$). So, we have seen this before, this is the probability that any other site misses this multiplied with itself nm times which is S = (1 - $\frac{1}{n})^{nm}$, which is e^{-m}. So, what we can see is as the traffic increases in the sense number of updates per site, as that increases, the residue decreases and ultimately the residue tends to 0.

So, if you just increase the traffic, then the residue will reduce, reduce, reduce, reduce, and ultimately it will come down to 0. And how will it come down? Well, it will come down using this exponential relationship over here.

(Refer Slide Time: 45:50)



Communication between Nodes
Epidemic Protocols
Gossip Based Protocols

Anti-Entropy
Rumor Mongering
Deleting Nodes and Spatial Distribution

Backup with Anti-entropy

- Rumor mongering can miss sites. (residue)
- After a certain time, we can run an anti-entropy protocol.
- If two sites discover a missing update, then they could start a hot rumor.
- Xerox Clearinghouse did some redistribution through direct mail also.

epidemic [Anti-entropy { push, pull ; Rumor mongering → residue e^{m}

Smruti R. Sarangi — Communication between Nodes

So, then, what is the key that we key learning that we got from here that rumor mongering can miss sites it clearly can, because there is a residue. After a certain time, we can run the anti-entropy protocol. So, what we can do is we can do rumor mongering for some time. And the rumor mongering is nice good it is self-terminating.

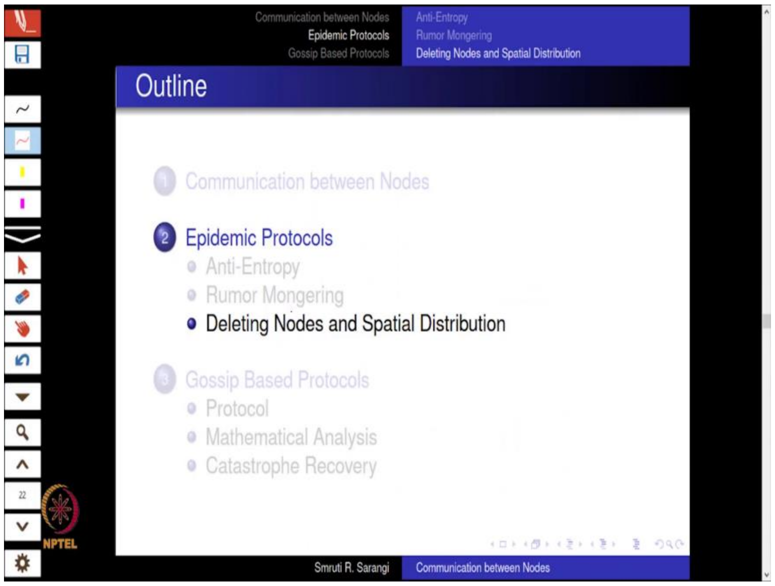
But after a certain time, we can again run a background kind of slow anti entropy protocol to ensure that all the nodes are genuinely covered. And let us say whenever they discover a missing update, you can also add a threshold to this, they can then start a hot rumor in their local region such that we can quickly cover their local region and the rumor itself will tie your fade out over time.

So, Xerox Clearinghouse did use many of these mechanisms. In addition, it did some amount of redistribution via direct mail also, this is something that we have not discussed. So, what we have discussed is primarily anti entropy there we said look, look at the beginning you follow a push based technique then towards the end you follow the pool based technique then we discuss rumor mongering.

So, we talked about the residue. I will also have said that look the residue is kind of e^{-m}. So, basically increase in traffic per site and the residue will quickly fall down to 0. So, the details are given in the paper of how exactly the residue and remove nodes etc are calculated.

But this should be enough to at least give you a feel of what are the broad mechanisms of these epidemic protocols. So, these are modeled on the lines of an epidemic. So, that is the reason they call them epidemic inspired protocols.

(Refer Slide Time: 48:12)



Communication between Nodes
Epidemic Protocols
Gossip Based Protocols

Anti-Entropy
Rumor Mongering
Deleting Nodes and Spatial Distribution

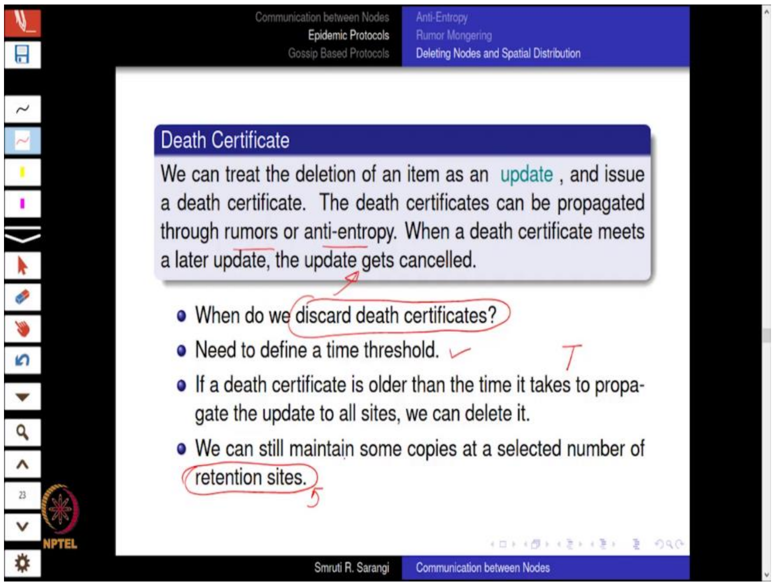
Outline

1. Communication between Nodes
2. Epidemic Protocols
    - Anti-Entropy
    - Rumor Mongering
    - Deleting Nodes and Spatial Distribution
3. Gossip Based Protocols
    - Protocol
    - Mathematical Analysis
    - Catastrophe Recovery

Smruti R. Sarangi Communication between Nodes

So, now let us look at other final aspects, deleting notes and spatial distribution.

(Refer Slide Time: 48:27)



Communication between Nodes
Epidemic Protocols
Gossip Based Protocols

Anti-Entropy
Rumor Mongering
Deleting Nodes and Spatial Distribution

Death Certificate

We can treat the deletion of an item as an update, and issue a death certificate. The death certificates can be propagated through rumors or anti-entropy. When a death certificate meets a later update, the update gets cancelled.

- When do we discard death certificates?
- Need to define a time threshold. T
- If a death certificate is older than the time it takes to propagate the update to all sites, we can delete it.
- We can still maintain some copies at a selected number of retention sites.

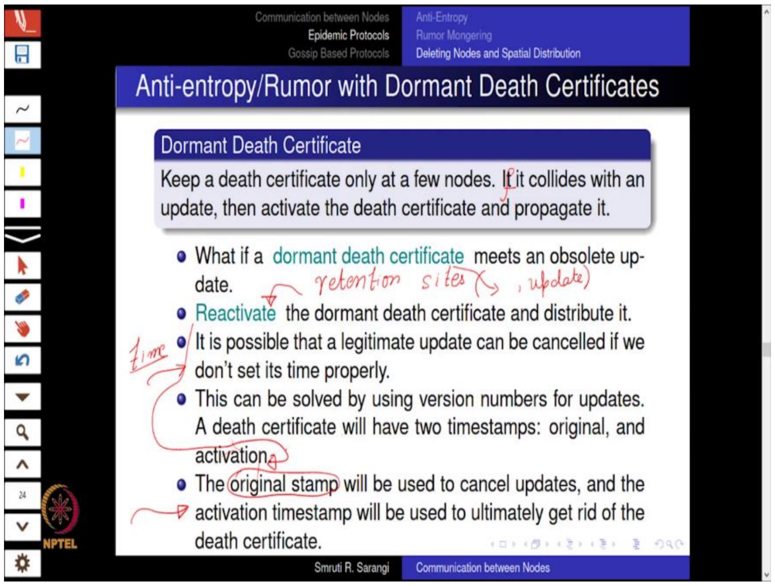
Smruti R. Sarangi Communication between Nodes

So, we can also treat the deletion of any item as an update the same way that when we modify an item we call an audit and update. We can also treat the deletion as an update and issue a death certificate for a node. The death certificates themselves can be propagated via rumors or anti entropy. And of course, when a death certificate meets a later update for that item, it basically means that that item, whichever site updated, the item did not get the death certificate. So then, of course, the update gets cancelled. But of course, we need to discard death certificates.

Well, if you are using rumor mongering, they will ultimately fade out. Otherwise, we define a time threshold. If a death certificate is older than the time it takes to propagate the update to all sites, we can delete it. So, essentially some sort of Max threshold. And if let us say the current time minus the time or the death certificate is more than that, we just delete it.

At some sites, we can still maintain called retention sites, we can still maintain the death certificates. So, is that later on, let us say a long, long time later, there is some update hiding somewhere, then that update can be checked against this death certificate. So, the retention sites can do that. So, something similar was also used in the Xerox protocol.

(Refer Slide Time: 49:48)



Communication between Nodes | Epidemic Protocols | Gossip Based Protocols | Anti-Entropy | Rumor Mongering | Deleting Nodes and Spatial Distribution

Anti-entropy/Rumor with Dormant Death Certificates

Dormant Death Certificate

Keep a death certificate only at a few nodes. If it collides with an update, then activate the death certificate and propagate it.

- What if a dormant death certificate meets an obsolete update. retention sites (x, update)
- Reactivate the dormant death certificate and distribute it.
- It is possible that a legitimate update can be cancelled if we don't set its time properly.
- This can be solved by using version numbers for updates. A death certificate will have two timestamps: original, and activation
- The original stamp will be used to cancel updates, and the activation timestamp will be used to ultimately get rid of the death certificate.

time

Smruti R. Sarangi | Communication between Nodes

So, now let us come to the idea of a dormant death certificate. So, this is that we keep a death certificate only at a few nodes. If it collides, there should be if it collides with an update, then of course, we activate the death certificate and propagate it, which is what we discussed in the previous slide, as well with regards to retention sites. So, what if a dormant dead certificate meets an obsolete update? In this case, we reactivate the dormant death certificate and distributed.

It is possible that a legitimate update can be canceled if we do not set its time properly. So, of course setting and managing the time is rather important. So, we can have this can be solved by using version numbers for updates. So, what will happen is that a death certificate can have two timestamps, so original timestamp, and an activation timestamp.

The original timestamp will be used to cancel updates. And the activation timestamp will be used to ultimately get rid of the death certificate, which means that if let us say the activated death certificate has been propagating for a long time, we can discard it. So, what is the broad idea? Well, the broad idea is that look, in an unstructured network, our messages can be floating around undetected.

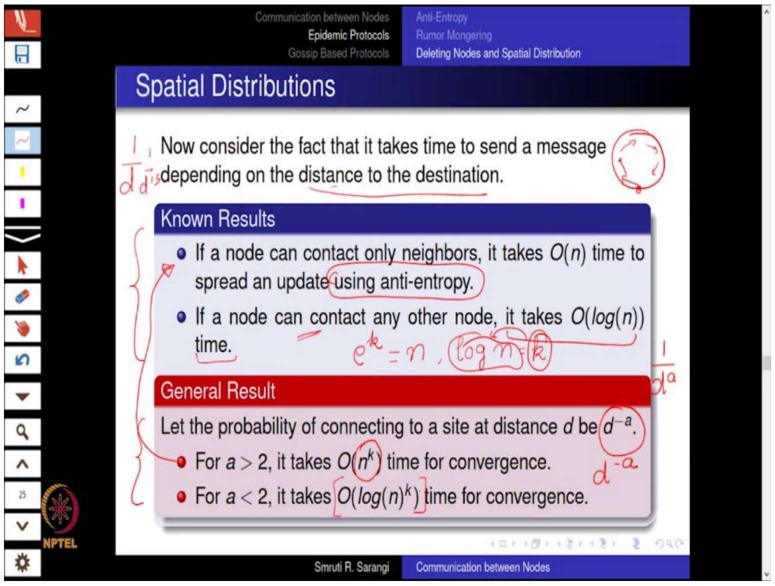
It is a large network, some node will have some message later on, it will wake up and start propagating the message. So if you delete a node, it might still not lead to an actual deletion. Because it is possible that maybe long time later some node will show up with a message.

If you delete not a node, but if you delete some data, a later update can always be there lurking around. So, that is the reason we had this notion of death certificates stored at retention sites. So they will have an original timestamp. And whenever there is a collision of the dormant death certificate with an update, you just look at the timestamp.

The timestamp is after the data died. Well you can sell the update. But then there is a need to reactivate the dormant death certificate and again re-circulate because it appears that there are several sites that have not gotten the original death certificate. Now, the question is what should be the time on it?

Do not change the original time that remains, but also have an additional time called the activation time. So, why is this time required? This time is required to see for how long has, this version of the death certificate been in circulation. If it has been in circulation for a long time to reduce the traffic in the network, we can just kill that dormant dead certificate and remove the messages.

(Refer Slide Time: 53:13)


Communication between Nodes | Epidemic Protocols | Gossip Based Protocols — Anti-Entropy | Rumor Mongering | Deleting Nodes and Spatial Distribution

Spatial Distributions

Now consider the fact that it takes time to send a message depending on the distance to the destination.

Known Results

- If a node can contact only neighbors, it takes $O(n)$ time to spread an update using anti-entropy.
- If a node can contact any other node, it takes $O(log(n))$ time.

$e^k = n, \log n = k$

General Result

Let the probability of connecting to a site at distance d be d^{-a}.

- For $a > 2$, it takes $O(n^k)$ time for convergence.
- For $a < 2$, it takes $O(log(n)^k)$ time for convergence.

Smruti R. Sarangi | Communication between Nodes

Now, a few more issues so we will discuss some results of course without proof, the proof is there in the paper and in the references. Consider the fact that it takes time to send a message depending upon the distance to the destination. Some known results again presented without proof. If a node can contact only its neighbors it takes order n time to spread an update using anti entropy So that should not come as a surprise.

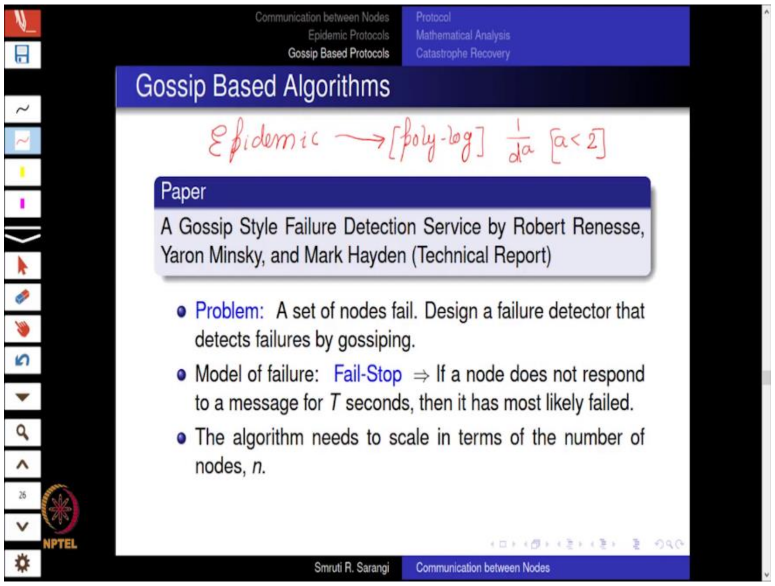
So, let us say if this is a ring, and let us say this one site, if I can only contact my neighbors, this is how I go and it will take order n time. However, if a node can contact any other node, it will take order of log n time to actually transmit the message using anti entropy. And this again should not come as a surprise because initially we just do a push-based system and a push-based system we reduce something by a constant factor.

And let us say if this factor is e, and let us say we have k search rounds. So, in every site we reduce the susceptibility by $\frac{1}{e}$. So, we can also approximate that to saying that we increase the number of susceptible sites by e, which again is not totally correct, but serves as an approximation kind of, and so, then we can say that for e^k= n, well, this means that the natural log with respect to e of n = k, which has a number of rounds. So, that is roughly how the term order of log n comes, but of course, we do push at the beginning pool at the end does not matter already, it just takes log n steps, you can kind of keep that in mind without going into the details of a rigorous proof that this is the time it takes anti entropy to reach convergence or roughly all the nodes. We can then kind of extend this with some specific results. So, let the probability of connecting to a site at distance

d. So, it is less distance be defined as the number of hops in the network let the probability of that be d^{-a}. So, this is basically d raised to the power some quantity fair is a > 2 which is basically it is an, a > 2.

So, this is kind of stronger than the inverse square law. So, it will take a long time to converge which is O(n^k) for convergence. And this kind of can be seen as a generalization of this result. On the other hand, if a < 2, which means it is kind of weaker than an inverse square law in the sense maybe it is $\frac{1}{d}$ or $\frac{1}{d^{1.5}}$ something like that, then it will take order a poly log time for convergence again a generalization of this result for convergence, it will take O(log$(n)^k$). So, that important, I am just moving to another slide to give myself some space.

(Refer Slide Time: 56:38)



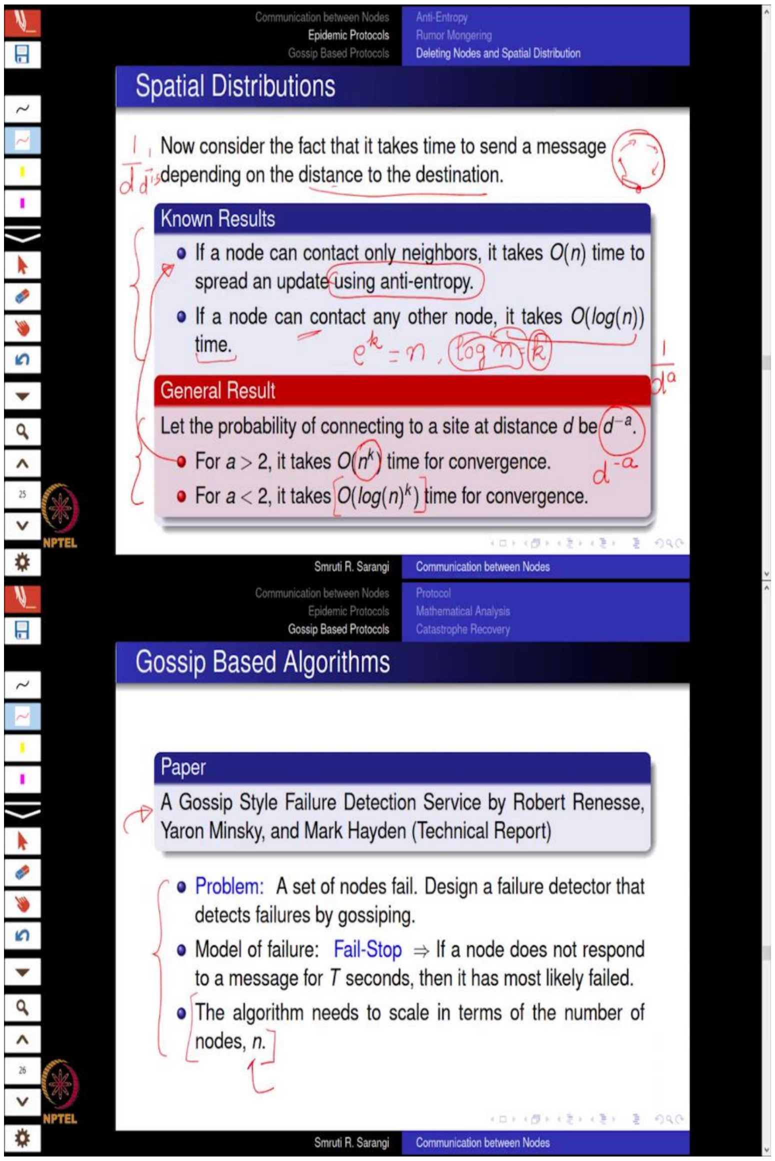
Gossip Based Algorithms

Epidemic $\longrightarrow$ [poly-log] $\frac{1}{d^a}$ [a < 2]

Paper

A Gossip Style Failure Detection Service by Robert Renesse, Yaron Minsky, and Mark Hayden (Technical Report)

- Problem: A set of nodes fail. Design a failure detector that detects failures by gossiping.
- Model of failure: Fail-Stop $\Rightarrow$ If a node does not respond to a message for T seconds, then it has most likely failed.
- The algorithm needs to scale in terms of the number of nodes, n.



Communication between Nodes | Epidemic Protocols | Gossip Based Protocols — Anti-Entropy | Rumor Mongering | Deleting Nodes and Spatial Distribution

Spatial Distributions

Now consider the fact that it takes time to send a message depending on the distance to the destination.

Known Results

- If a node can contact only neighbors, it takes $O(n)$ time to spread an update using anti-entropy.
- If a node can contact any other node, it takes $O(log(n))$ time.

$e^k = n$, $\log n = k$

General Result

Let the probability of connecting to a site at distance d be d^{-a}.

- For $a > 2$, it takes $O(n^k)$ time for convergence.
- For $a < 2$, it takes $O(log(n)^k)$ time for convergence.

$\frac{1}{d}$ $\frac{1}{d^{1.5}}$ $\frac{1}{d^a}$ d^{-a}

Smruti R. Sarangi | Communication between Nodes

Communication between Nodes | Epidemic Protocols | Gossip Based Protocols — Protocol | Mathematical Analysis | Catastrophe Recovery

Gossip Based Algorithms

Paper

A Gossip Style Failure Detection Service by Robert Renesse, Yaron Minsky, and Mark Hayden (Technical Report)

- Problem: A set of nodes fail. Design a failure detector that detects failures by gossiping.
- Model of failure: Fail-Stop $\Rightarrow$ If a node does not respond to a message for T seconds, then it has most likely failed.
- The algorithm needs to scale in terms of the number of nodes, n.

Smruti R. Sarangi | Communication between Nodes

So, the important point over here. Just a continuation of the previous slide. The important point is that look in any epidemic based algorithm if you want to achieve a poly-log time convergence, which means the message reaches everybody particularly using anti entropy, then you need to contact random nodes with a probability. So, this basically means that just contacting neighbors and so on is not good enough. There should be some facility of contacting even faraway neighbors with a reasonably decent probability so that the epidemic can spread faster.

So, coronavirus, does not work that way and Coronavirus is still a neighbor to neighbor transmission will depending upon how neighbors are, because so that is the reason it has not engulfed the entire world yet.

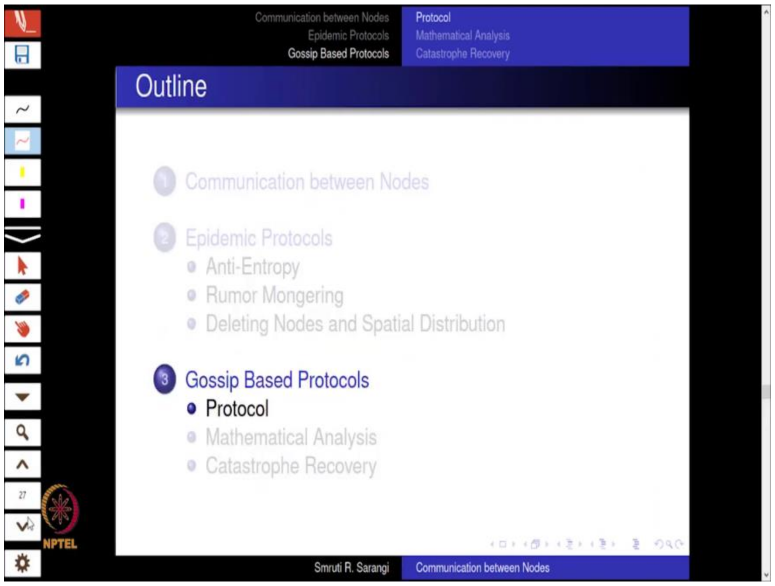
But in an epidemic kind of scenario. If we really want to reach everybody, then a < 2 and if we want to restrict it a > 2. So, this is when from a poly log it will become a polynomial.

So, we did take some space out of this slide to kind of discuss the results of epidemic based distribution. So, now, what we will do is first apologize for using the real estate of the gossip algorithms slide and then discuss gossip based algorithms. So, gossip based algorithms are of a different kind, they are of a different nature, they are not based on epidemics.

So, the key reference that we will use is actually this paper a gossip style failure detection service and this is one of the seminal references in this area. So, it is that if a set of nodes fail design a failure detector that detects failure by gossiping. So, gossiping is basically that, you just pass on that information to whoever you know.

So, the model of failure is fail stop, which means, if a node does not respond to a message for T seconds, we see that it has most likely failed. And the important thing is similar to rumor mongering and anti-entropy with n algorithm needs to scale. So, we will look at such gossip based mechanisms.

(Refer Slide Time: 59:31)

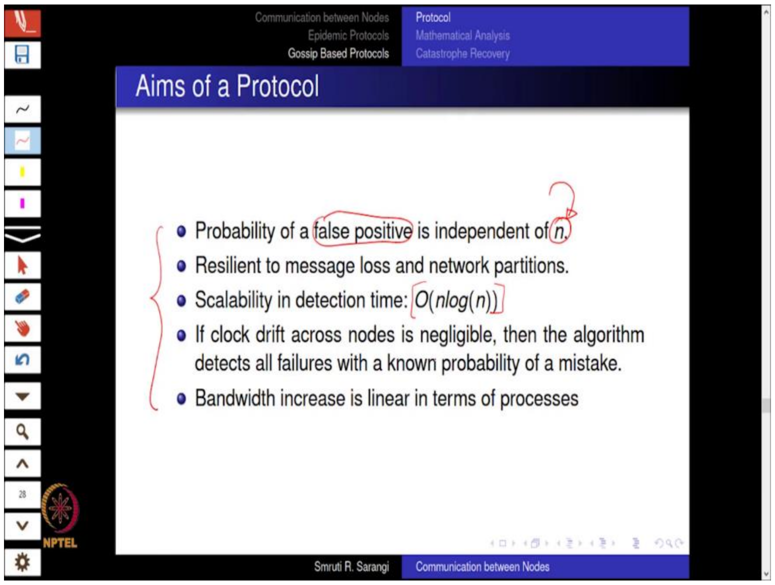

Communication between Nodes
Epidemic Protocols
Gossip Based Protocols

Protocol
Mathematical Analysis
Catastrophe Recovery

Outline

1. Communication between Nodes
2. Epidemic Protocols
   - Anti-Entropy
   - Rumor Mongering
   - Deleting Nodes and Spatial Distribution
3. Gossip Based Protocols
   - Protocol
   - Mathematical Analysis
   - Catastrophe Recovery

Smruti R. Sarangi Communication between Nodes


Communication between Nodes
Epidemic Protocols
Gossip Based Protocols

Protocol
Mathematical Analysis
Catastrophe Recovery

Aims of a Protocol

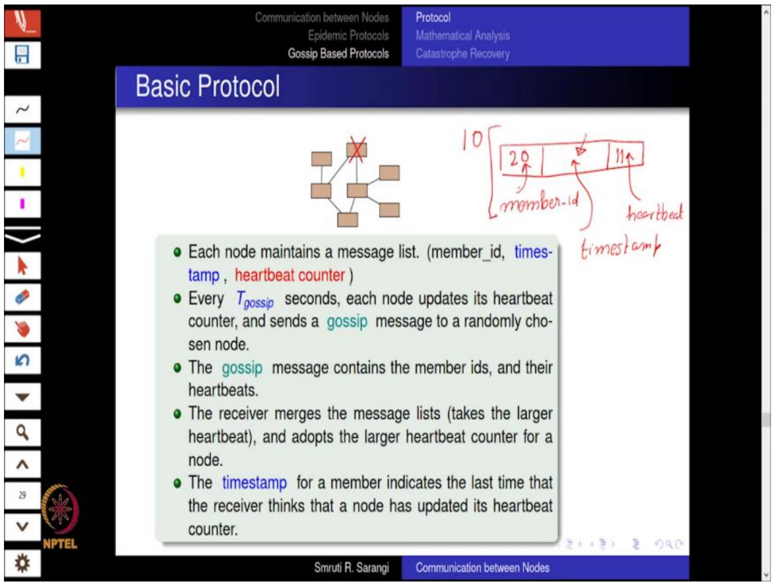
- Probability of a false positive is independent of n.
- Resilient to message loss and network partitions.
- Scalability in detection time: $\boxed{O(nlog(n))}$
- If clock drift across nodes is negligible, then the algorithm detects all failures with a known probability of a mistake.
- Bandwidth increase is linear in terms of processes

Smruti R. Sarangi   Communication between Nodes

Next will now discuss the gossip protocol. So, the aims of the protocol are as follows. The first is that the probability of a false positive which means that we say that a given node has a failure, but actually it is not failed. This is the probability of a false positive, this is independent of n. So, this is a rather strong assumption, or rather a strong aim in the sense that regardless of the scalability of the network, our probability of a false positive is bounded and it is not dependent on n. Furthermore, our protocol is resilient to message loss and network partitions. So, regardless of the fact that, networks and partition messages or loss protocols can still function.

And the scalability in detection time is achieved. In the sense the time it takes to detect all failures is order of n login. And if the clock drift across the nodes is negligible, then the algorithm detects all failures with a known mistake probability. And the increase in bandwidth, which is pretty much the number of messages is linear in terms of the number of processes. So, the key operative point here is that the probability of false positive is bounded. The detection time is O(nlog(n)) roughly linear and the bandwidth is linear.

(Refer Slide Time: 1:01:02)



Basic Protocol

- Each node maintains a message list. (member_id, timestamp, heartbeat counter)
- Every T_{gossip} seconds, each node updates its heartbeat counter, and sends a gossip message to a randomly chosen node.
- The gossip message contains the member ids, and their heartbeats.
- The receiver merges the message lists (takes the larger heartbeat), and adopts the larger heartbeat counter for a node.
- The timestamp for a member indicates the last time that the receiver thinks that a node has updated its heartbeat counter.

So, what we do is that each node maintains a message list, write a list of messages. And each of the items in the message list contains three fields. The fields are member id, timestamp and heartbeat counter. So, Member id is clearly some information about another node. So, the idea of that two node is being called Member id.

So, what we do is that every T_{gossip} seconds each node will update its heartbeat, increase its heartbeat counter and send a gossip message to a randomly chosen node something similar to anti entropy is done here.

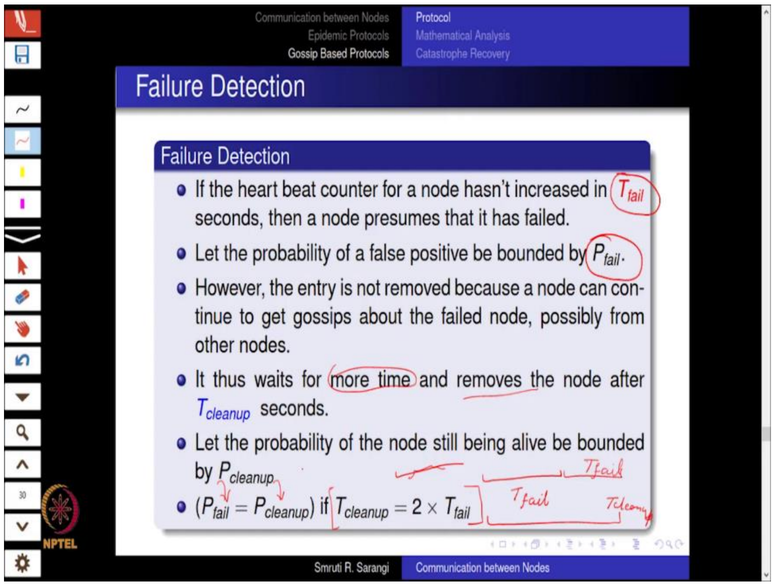
So, the heartbeat is a monotonically increasing counter and an increased heartbeat tells the rest of the nodes that it is alive. The Gossip message contains the member IDs and their heartbeats. So, basically the gossip message can also include the message list which is the Member id timestamp and heartbeats of all the information that a given node has.

The receiver will merge the message lists which basically means that for the same member id it will take the larger heartbeat and it will adopt the larger heartbeat counter for that node. Finally, the timestamp for the member indicates the last time that the receiver thinks that denote has updated its heartbeat counter. So, let us say that the heartbeat counter for node number 20 fer this message list is a part of known number 10. It is heartbeat currently is 11.

So, the timestamp for this entry essentially indicates the last time that the receiver thinks that node 20, updated its heartbeat count to 11. If in a world where clocks are perfectly synchronized, the timestamp can be a global timestamp in the sense that whenever a node increases its heartbeat. In all messages, it can just attach its timestamp and it can send the message then all nodes will, since they uniformly agree on the time, they can, they will be sure that look at a certain timestamp.

Node number 20 incremented its heartbeat. But in a world where timestamps themselves are not synchronized, there is no clock synchrony. The timestamp itself is an approximate quantity. So, the timestamp for a given member will indicate at best an approximate estimate of when the heartbeat was updated. So, the way that we will set the timestamp is that whenever we receive a message about a member, we will you know directly from the member we will set the timestamp to that right the time of receiving a message if their clocks are not synchronized.

(Refer Slide Time: 1:04:23)



Failure Detection

Failure Detection

- If the heart beat counter for a node hasn't increased in T_{fail} seconds, then a node presumes that it has failed.
- Let the probability of a false positive be bounded by P_{fail}.
- However, the entry is not removed because a node can continue to get gossips about the failed node, possibly from other nodes.
- It thus waits for more time and removes the node after $T_{cleanup}$ seconds.
- Let the probability of the node still being alive be bounded by $P_{cleanup}$.
- $(P_{fail} = P_{cleanup})$ if $T_{cleanup} = 2 \times T_{fail}$

So, how do we detect if failure? Will detecting a failure again is an approximate activity say the heartbeat counter for a node has not increased in T_{fail} seconds, then a node is presumed to have failed. So, let us assume that for whatever reason, either messages got delayed or there was a partition in the network or the receiver had crashed and then it recovered.

There is a probability of a false positive and this probability of false positive in this case is bounded by P_{fail} which basically what is false positive we think a node has failed, but actually it has not failed, then what will happen is that we will incorrectly conclude that the node has failed. However, the entry in the message list is not removed, because a node can continue to get gossips about the failed node from other nodes right it can continue to get gossips.

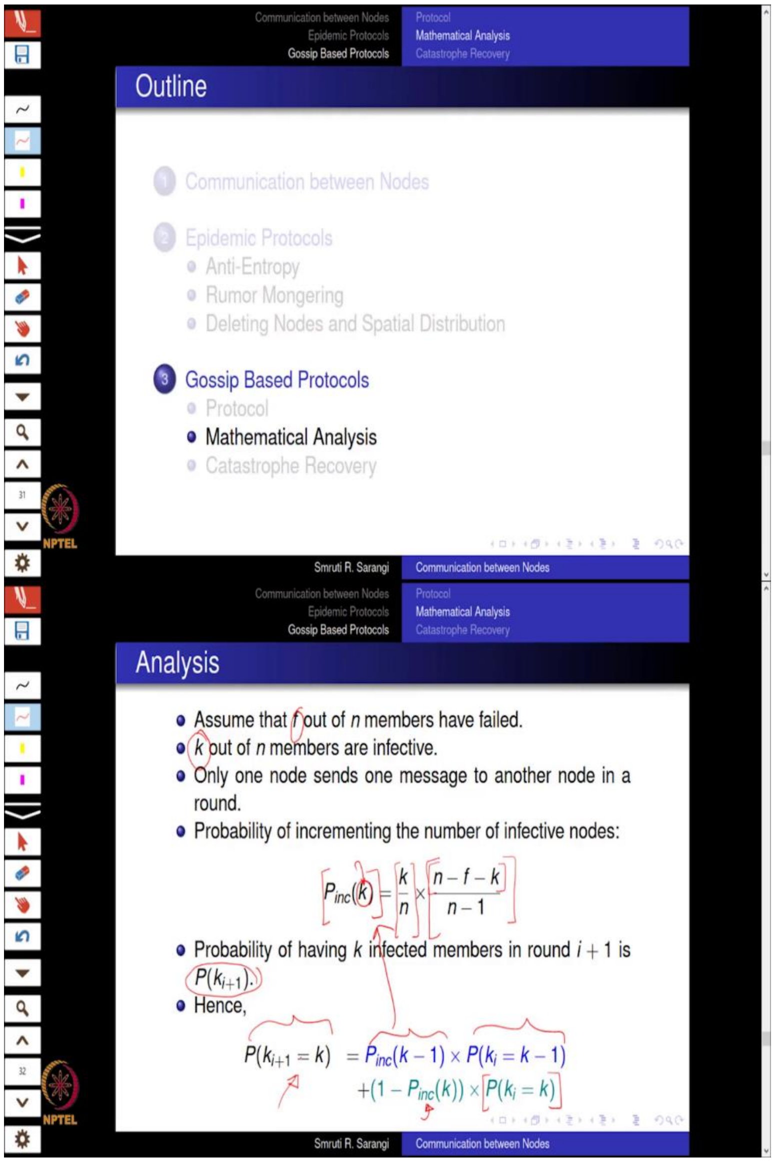
So even after it concludes, it has failed it will not remove the entry it will rather keep it. So, it does wait it does waits for some more time and finally removes the node after a $T_{cleanup}$ seconds. So let the probability of the node still being alive be bounded by $P_{cleanup}$ the probability that after $T_{cleanup}$ seconds the node is still alive.

Another variant of the false positive so both of these probabilities P_{fail} = $P_{cleanup}$ is equal to. So, whenever both of these probabilities equal. So, we can see that, after T_{fail} seconds, we conclude that look node has failed. But again, there is a mistake probability in the sense there is a probability of false positive of P_{fail} . After that, we wait for some more time. So, this entire duration, we can call it as $T_{cleanup}$.

So, the question is that given the fact that we have come here, and in spite of that, if the node, we again, wait for this much of time, in spite of waiting for this much of time, if the node is still not responsive, can we conclude it as failed? Maybe Yes, We can and we can remove the list, and we can remove it from the list, but the probability of a false positive in this interval, which is again, conditioned on the fact that, we waited for T_{failed} seconds, and we did not hear anything about the node.

If we want this probability $P_{cleanup}$ = P_{fail} then it is obvious that this interval should also be equal to T_{fail} . Hence, $T_{cleanup}$ = 2 X T_{fail} . So that is when both of these probabilities will be equal. But as I said, there is no specific need of making them equal other than for some degree of mathematical elegance. So, you can refer to the paper, but in general, having mathematical elegance while analyzing such complex systems is desirable. And to do that, all that we need to do is we need to set the $T_{cleanup}$ = 2 X T_{fail} .

(Refer Slide Time: 1:08:01)



Outline

1. Communication between Nodes
2. Epidemic Protocols
   - Anti-Entropy
   - Rumor Mongering
   - Deleting Nodes and Spatial Distribution
3. Gossip Based Protocols
   - Protocol
   - Mathematical Analysis
   - Catastrophe Recovery

Smruti R. Sarangi | Communication between Nodes

Analysis

- Assume that f out of n members have failed.
- k out of n members are infective.
- Only one node sends one message to another node in a round.
- Probability of incrementing the number of infective nodes:

$$P_{inc}(k) = \frac{k}{n} \times \frac{n-f-k}{n-1}$$

- Probability of having k infected members in round $i+1$ is $P(k_{i+1})$.
- Hence,

$$P(k_{i+1} = k) = P_{inc}(k-1) \times P(k_i = k-1) + (1 - P_{inc}(k)) \times P(k_i = k)$$

Smruti R. Sarangi | Communication between Nodes

Let us do some more math. So, let us assume that f out of n members are failed. So, we need to detect all f, k out of n members are infective. So, infective means that they are spreading the message about failed nodes. And the third point is only one node sends one message to another node in the round. So, in a given round, only one node sends one message to another node in a round. So, what is the probability of incrementing the number of infective nodes? So, the probability of incrementing it let us say that the current number of infective nodes is $P_{inc}(K)$, the probability of infecting it is given by this expression.

So, let us work it out. So, the first factor is the probability that one of the infective nodes actually sends a message and this is $\frac{k}{n}$, because we are assuming a uniform probability of sending a message. Next, what is the probability of a node that is not failed and not infected getting the message.

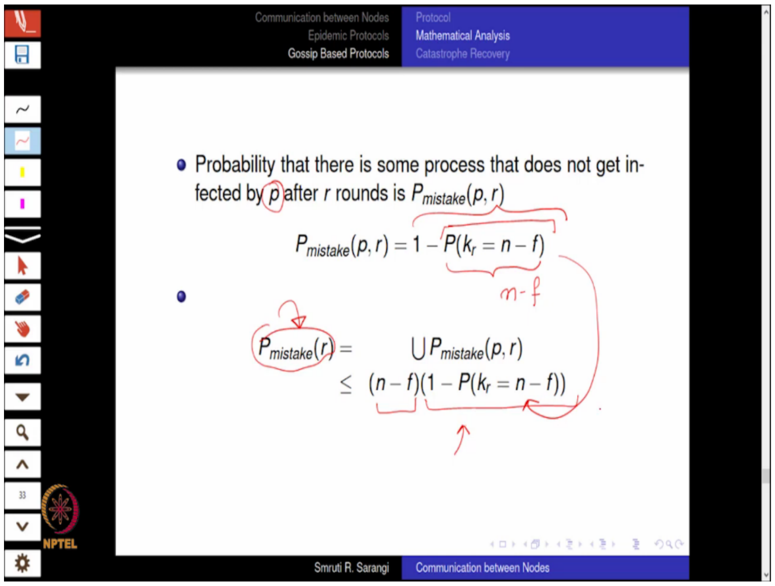
So, the number of such desirable nodes for the sake of this probability calculation is n minus the number of failed nodes minus the number of infective nodes and n- f- k. And of course, there are n -1 choices, so, the total sample space is n-1 and thus the probability is this and since they are independent events, they get multiplied.

So, the probability of having k infected members in round i plus one let it be P($k_i + 1$). So, the P($k_i + 1$) = k we can infer the following recursive relationship. So, here I would like to make a point that whenever a system is complicated and difficult to handle, and we cannot come up with a closed form expression, we often try to set up some sort of a recurrence relation and recurrence is related to recursion, where essentially is the same expression repeating over and over again.

So, in such kind of a scenario, where we have set up a recurrence, what we can see is that in the (i +1) it round if we have k infected members, in a sense k member who have some information about the number of failed nodes. So, this is equal to either $P_{inc}(K - 1)$ X P(k -1) which means i th round we had k - 1 infective members, and then there was an increment and the increment is given by this expression.

Or in the i th round, we already had k infective members, and there was no increment which is given by this expression. So, both of them can be calculated. And we can use this recurrence relation to nicely calculate the P $(k_i = k)$, this can be done.

(Refer Slide Time: 1:11:21)



- Probability that there is some process that does not get infected by p after r rounds is $P_{mistake}(p, r)$

$$P_{mistake}(p, r) = 1 - \underbrace{P(k_r = n - f)}_{n-f}$$

-

$$P_{mistake}(r) = \bigcup P_{mistake}(p, r)$$
$$\leq (n - f)(1 - P(k_r = n - f))$$

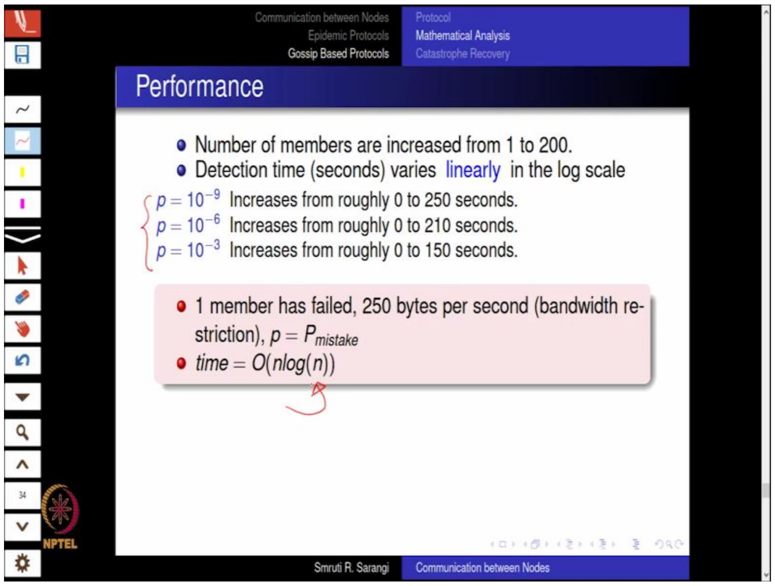
So, the probability that there is some process that does not get infected by let us say process P after r rounds, let it be $P_{mistake}$(P,r). So, we will use this expression this probability P over here. So, $P_{mistake}$(P,r) = 1- P(k_r = n – f). So, let us understand this slowly.

So, let us first break this expression down into its sub components and understand each of these components one after the other. So, the first thing says that look at the end of r rounds, what is the probability that (n – f) nodes are infective. So, what where does (n – f) come from? n is the total number of nodes, f is the number of failed nodes.

So, (n – f) refers to all the nodes that have not failed. So, if all of those nodes at the end of the r th round are infective, this probability is given by P(kr = n – f). So, if at least one of the nodes right that is it there is at least one. so, some process means at least one process is still infected is given by $P_{mistake}$ (P, r) = 1 – P(kr = n – f).

And so, then this was for a given process P. If I want to look at all the processes, well, then I can use this simple result of probability theory, where $P_{mistake}$ (r) = ∪ $P_{mistake}$ (P, r) which is kind of ≤ it is upper bounded by. So, $P_{mistake}$ (P, r) is given by this expression over here and it is upper bounded by the total number of non failed nodes which is(n – f) times this quantity over here. So, which means that we have an upper bound for $P_{mistake}$ (r), which is this expression. So, let is keep moving.

(Refer Slide Time: 1:13:49)



Performance

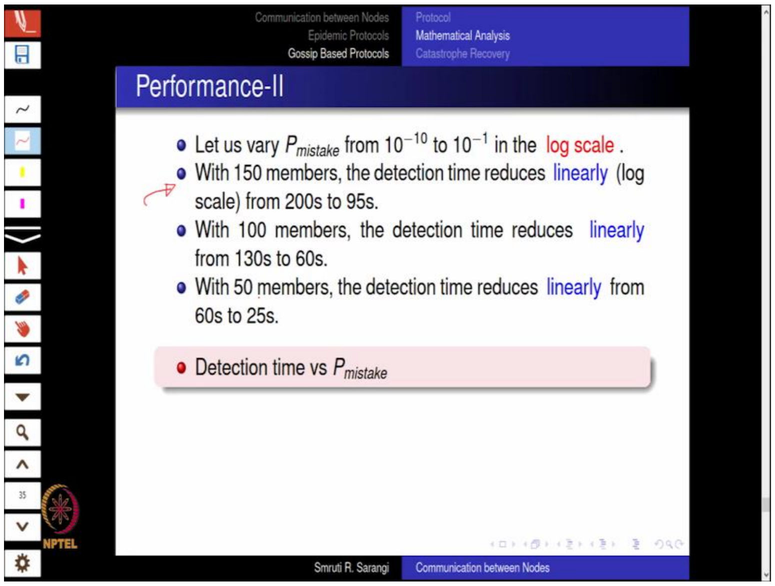
- Number of members are increased from 1 to 200.
- Detection time (seconds) varies linearly in the log scale

$p = 10^{-9}$ Increases from roughly 0 to 250 seconds.
$p = 10^{-6}$ Increases from roughly 0 to 210 seconds.
$p = 10^{-3}$ Increases from roughly 0 to 150 seconds.

- 1 member has failed, 250 bytes per second (bandwidth restriction), $p = P_{mistake}$
- $time = O(nlog(n))$

So, let us discuss the performance results. So, the experiment that was performed in this paper is the number of members this is this number is increased from 1 to 200. So, what we see is that the failure detection time varies linearly in the log scale. So, it is the probability is 10^{-9} increases from roughly 0 to 250 seconds 10^{-6} from 0 to 10 and 10^{-3}, 0 to 150. So, if one member is failed, so, the bandwidth requirement is 250 bytes per second and the time = O(nlog (n)).

(Refer Slide Time: 1:14:38)



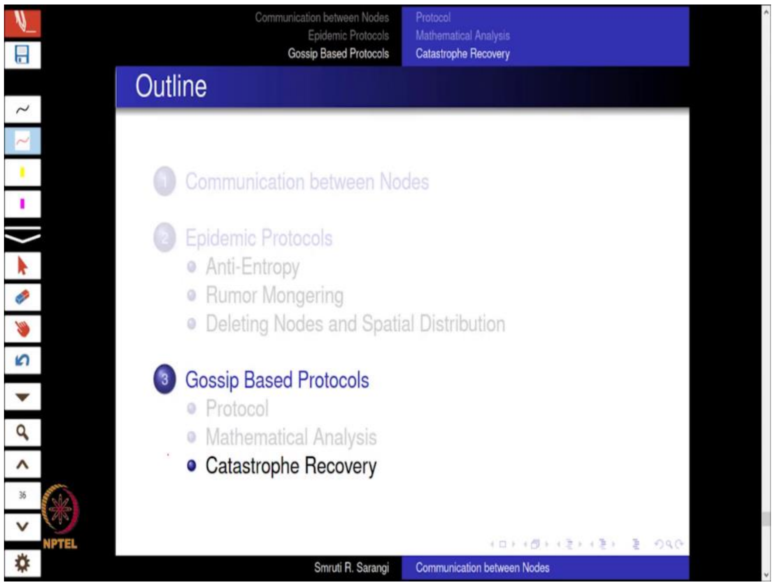
Performance-II

- Let us vary $P_{mistake}$ from 10^{-10} to 10^{-1} in the log scale .
- With 150 members, the detection time reduces linearly (log scale) from 200s to 95s.
- With 100 members, the detection time reduces linearly from 130s to 60s.
- With 50 members, the detection time reduces linearly from 60s to 25s.

- Detection time vs $P_{mistake}$

So, if let us say we vary the mistake, the mistake probability right that we derived two slides ago if we vary the mistake from 10^{-10} to 10^{-1} on a log scale, what we see is for that 150 members the detection time reduces linearly in the log scale from 200 seconds to 95 seconds with 100 members the detection time again reduces linearly from 130 to 160. And with 50 members it again reduces linearly from 60 to 25 seconds. So, this is the essentially detection time versus the mistake probability the mistake probability is something that we are willing to tolerate.

(Refer Slide Time: 1:15:25)

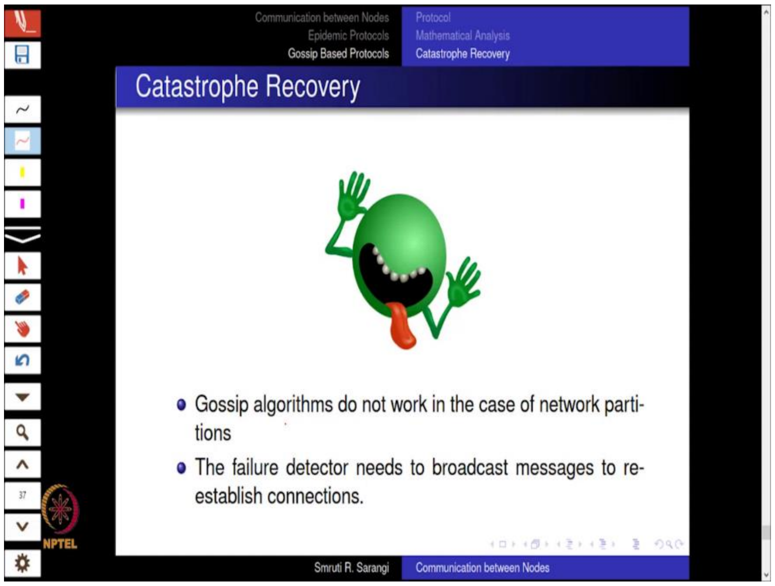


Communication between Nodes | Protocol
Epidemic Protocols | Mathematical Analysis
Gossip Based Protocols | Catastrophe Recovery

Outline

1. Communication between Nodes
2. Epidemic Protocols
   - Anti-Entropy
   - Rumor Mongering
   - Deleting Nodes and Spatial Distribution
3. Gossip Based Protocols
   - Protocol
   - Mathematical Analysis
   - Catastrophe Recovery

Smruti R. Sarangi | Communication between Nodes



Communication between Nodes | Protocol
Epidemic Protocols | Mathematical Analysis
Gossip Based Protocols | Catastrophe Recovery

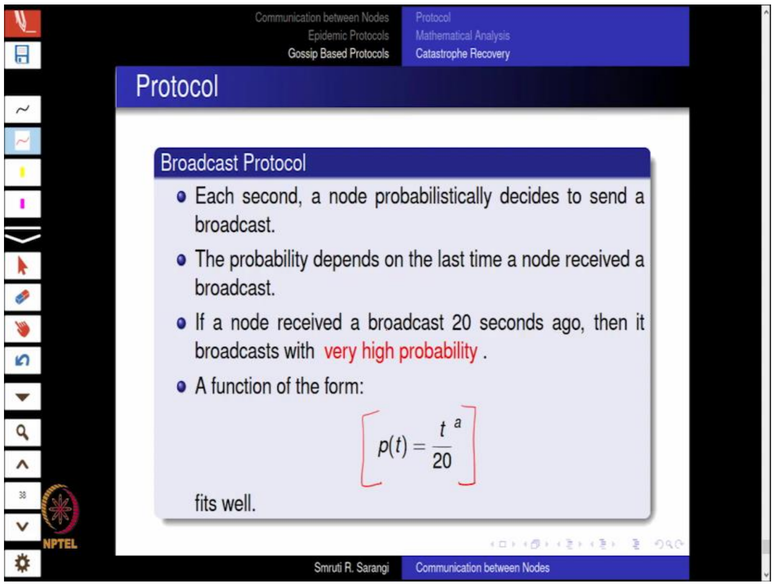
Catastrophe Recovery

- Gossip algorithms do not work in the case of network partitions
- The failure detector needs to broadcast messages to re-establish connections.

Smruti R. Sarangi | Communication between Nodes

Now, let us discuss catastrophe recovery. So, gossip algorithms where essentially what we are doing over here is we are gossiping, they do not work in the case of network partitions. So, the so a dedicated failure detector needs to broadcast messages and it needs to reestablish connections. So, unless you reestablish connections messages will not go to the part of the network that has been partitioned.

(Refer Slide Time: 1:15:55)



Protocol

Broadcast Protocol

- Each second, a node probabilistically decides to send a broadcast.
- The probability depends on the last time a node received a broadcast.
- If a node received a broadcast 20 seconds ago, then it broadcasts with very high probability .
- A function of the form:

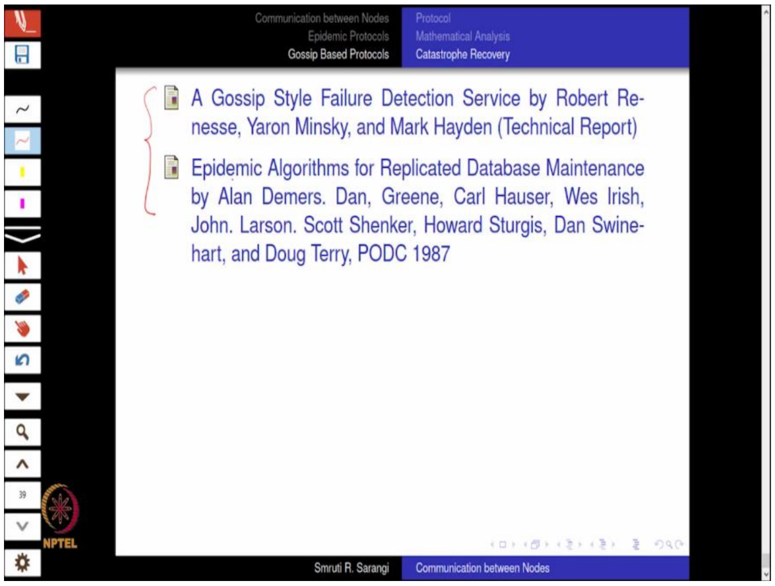
$$\left[p(t) = \frac{t}{20}^{a} \right]$$

fits well.

So, we can have a broadcast protocol where each second and node probabilistically decides to send a broadcast. This probability depends on the last time a node received a broadcast. So, let us say if a node received a broadcast 20 seconds ago, then it broadcasts with a very high probability.

And one of the functions of this form p(t) = $\frac{t^a}{20}$, which fits well. So basically, the idea is that if let us say a node has been receiving broadcasts from other nodes, it can reasonably be sure that the network is connected. But let us say a long time has elapsed and it has not gotten any broadcast, then it tries to re-establish connections.

(Refer Slide Time: 1:16:41)



- A Gossip Style Failure Detection Service by Robert Renesse, Yaron Minsky, and Mark Hayden (Technical Report)
- Epidemic Algorithms for Replicated Database Maintenance by Alan Demers. Dan, Greene, Carl Hauser, Wes Irish, John. Larson. Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry, PODC 1987

So, these were the broadly speaking the two papers that we discussed in this lecture. So, the first was epidemic algorithms. So, of course, it is mentioned in the reverse order. And the second was a gossip style failure detection service.