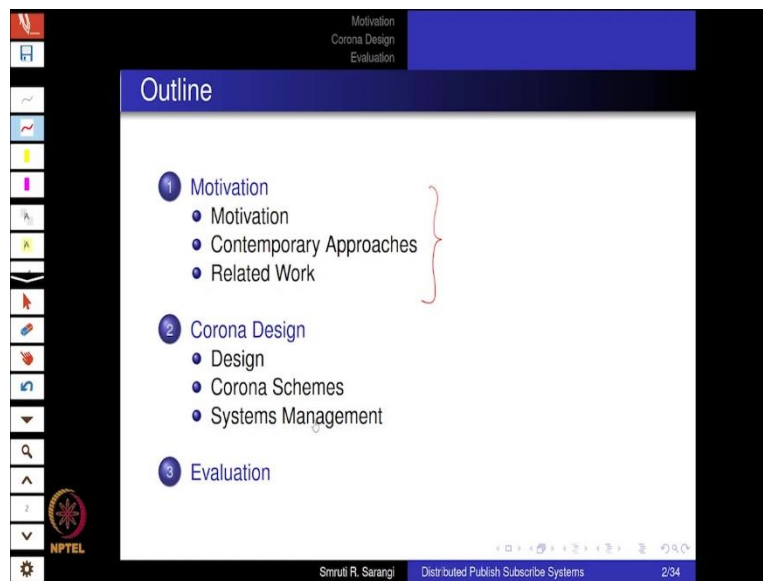


Advanced Distributed System
Professor Smriti R. Sarangi
Department of Computer Science and Engineering
Indian Institute of Technology, Delhi
Lecture 20
Corona: Distributed publish-subscribe system

In this lecture, we will describe the corona publish subscribe system. So, this is an extension of what we have been studying in the distributed with regards to distributed hash tables, particularly this is an extension of pastry. So, we will see how to use it in this setting.

(Refer Slide Time: 0:37)



The screenshot shows a presentation slide titled "Outline" with a blue header. The slide content is as follows:

- 1 Motivation
 - Motivation
 - Contemporary Approaches
 - Related Work
- 2 Corona Design
 - Design
 - Corona Schemes
 - Systems Management
- 3 Evaluation

A red bracket is drawn on the slide, grouping the three sub-points under "1 Motivation".

The slide footer contains the text "Smriti R. Sarangi Distributed Publish Subscribe Systems 2/34".

So, first we will describe the general structure of publish subscribe systems. Then we will discuss the design of Corona and finally, we will discuss the results the evaluation results of Corona.

(Refer Slide Time: 0:58)

Motivation

- The world wide web has a lot of information that changes frequently.
- There is no well defined publish-subscribe interface.
 - **Publish** : Post updates, and send to all the subscribers.
 - **Subscribe** : Register to get updates.
- Polling based methods are not efficient.
- Corona provides a scalable method to disseminate updates.

Smruti R. Sarangi | Distributed Publish-Subscribe Systems | 4/34

So, let us first describe the broad structure of a publish subscribe system. So, the broad structure is like this.

(Refer Slide Time: 1:11)

Pub-Sub system

Subscribers

- Express interest
- Notify the Pub-Sub system

RSS/Atom

Publishers

- Register
- Publish updates

• Stock prices

• news feeds

0:05:42

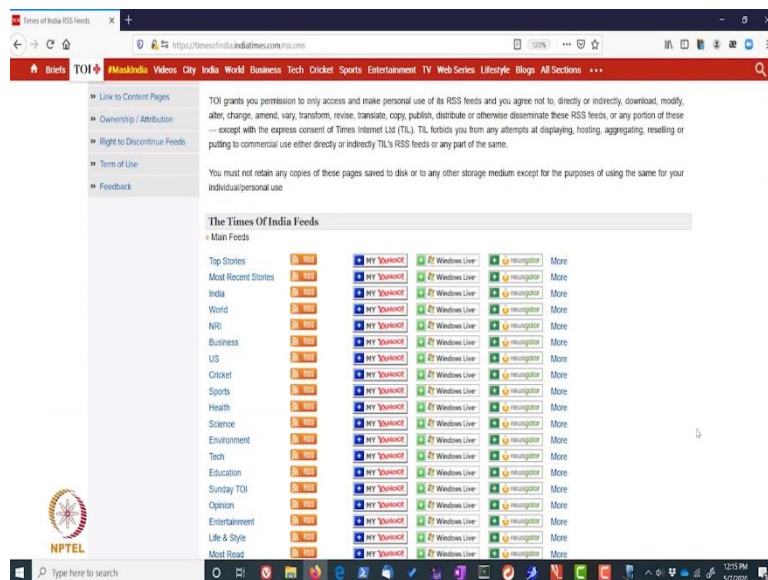
So, in any public subscribe system, we have a set of publishers and we have a set of subscribers. So, let us say that this is a hypothetical space. So, all of the publishers publish topics of interest publish stories of interest to the Pub Sub system so, what could this be? Well, this could be something like this, that so, let us consider the stock market. So, if I am a broker, I might be interested in let us say the stocks or 5 different companies.

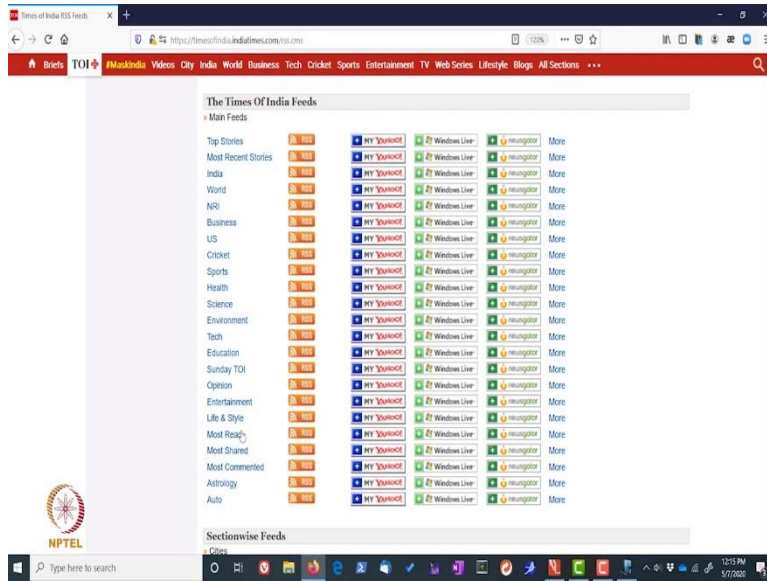
So, then whenever the stocks whenever the stock of any company changes, so, some dedicated server can just publish the updates to a pub sub system there might be a set of subscribers who would be interested in a subset of the stocks. So, the moment that there is a change a message will go from the Pub Sub system to the subscribers whichever subscriber is interested telling them that look there is a change that is an update. So, in the case of a stock market, one of the stocks that you were interested in that stock has changed. So, you would like to take a look at the new stock price.

So, in general what happens for subscribers is that they express interest in certain topics. So, this is notified to the Pub Sub system similarly, the publishers also the first register with the Pub Sub system and then they publish their updates so, this can be anything this can be stock feeds, stock prices, these can be news feeds. So, it can be anything so, if you will see one of the most common Pub Sub systems is called an RSS system.

So, RSS or atoms you would see RSS or Atom links with popular news websites. So, in this case what can happen is that I can take RSS feed reader like aggregator for example. And then whenever there is an RSS update that that will be shown to me. So, whenever there is a new news story that will be shown to me.

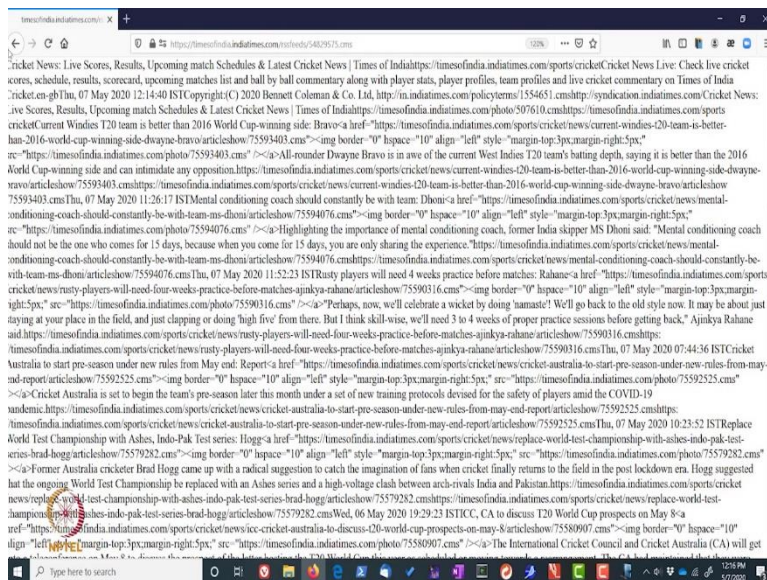
(Refer Slide Time: 5:01)





So, I am showing you the RSS page of Times of India is a very popular newspaper in India. So, if you see, So, the Times of India is publishing stories with respect to many topics. So, top stories India World Business cricket. So, if I want, I can subscribe to any of these topics, the moment that there is a new story in let us say, for example, science, I will have a dedicated RSS feed reader. And the dedicated RSS feed reader will show me the updated sites. So, let me click cricket, for example.

(Refer Slide Time: 5:36)



Times of India RSS feeds

English | Epaper | GaegonNow

THE TIMES OF INDIA

Briefs TOI+ #MasKIndia Videos City India World Business Tech Cricket Sports Entertainment TV Web Series Lifestyle Blogs All Sections

What is RSS?
 Times of India Feeds
 How to Use the Feeds
 Use of RSS Feeds
 Link to Content Pages
 Ownership / Attribution
 Right to Discontinue Feeds
 Terms of Use
 Feedback

RSS is a way of providing content to the user's browser or desktop in an efficient way. By using RSS feeds, the user can stay updated on the news from TOI and other news sources with little extra effort.

RSS (Really Simple Syndication) feeds are normally provided in three ways: headlines only, headlines with excerpts and full text feeds. TOI provides you headlines with excerpts, for free.

TOI grants you permission to only access and make personal use of its RSS feeds and you agree not to, directly or indirectly, download, modify, alter, change, amend, vary, transform, revise, translate, copy, publish, distribute or otherwise disseminate these RSS feeds, or any portion of these — except with the express consent of Times Internet Ltd (TIL). TIL forbids you from any attempts at displaying, hosting, aggregating, reselling or putting to commercial use either directly or indirectly TIL's RSS feeds or any part of the same.

You must not retain any copies of these pages saved to disk or to any other storage medium except for the purposes of using the same for your individual/personal use.

The Times Of India Feeds

Main Feeds

Top Stories	3,100	NY YouKnow	Windows Live	neungattar	More
Most Recent Stories	3,100	NY YouKnow	Windows Live	neungattar	More
India	3,100	NY YouKnow	Windows Live	neungattar	More
World	3,100	NY YouKnow	Windows Live	neungattar	More
NRI	3,100	NY YouKnow	Windows Live	neungattar	More
Business	3,100	NY YouKnow	Windows Live	neungattar	More
US	3,100	NY YouKnow	Windows Live	neungattar	More

Times of India RSS feeds

English | Epaper | GaegonNow

THE TIMES OF INDIA

Briefs TOI+ #MasKIndia Videos City India World Business Tech Cricket Sports Entertainment TV Web Series Lifestyle Blogs All Sections

Sports
 Health
 Science
 Environment
 Tech
 Education
 Sunday TOI
 Opinion
 Entertainment
 Life & Style
 Most Read
 Most Shared
 Most Commented
 Astrology
 Auto

Sectionwise Feeds

Cities

Mumbai	3,100	NY YouKnow	Windows Live	neungattar	More
Delhi	3,100	NY YouKnow	Windows Live	neungattar	More
Bangalore	3,100	NY YouKnow	Windows Live	neungattar	More
Hyderabad	3,100	NY YouKnow	Windows Live	neungattar	More
Chennai	3,100	NY YouKnow	Windows Live	neungattar	More
Ahmedabad	3,100	NY YouKnow	Windows Live	neungattar	More
Alahabad	3,100	NY YouKnow	Windows Live	neungattar	More
Bhubaneswar	3,100	NY YouKnow	Windows Live	neungattar	More
Coimbatore	3,100	NY YouKnow	Windows Live	neungattar	More
Gurgaon	3,100	NY YouKnow	Windows Live	neungattar	More
Gaunahati	3,100	NY YouKnow	Windows Live	neungattar	More

Times of India RSS feeds

English | Epaper | GaegonNow

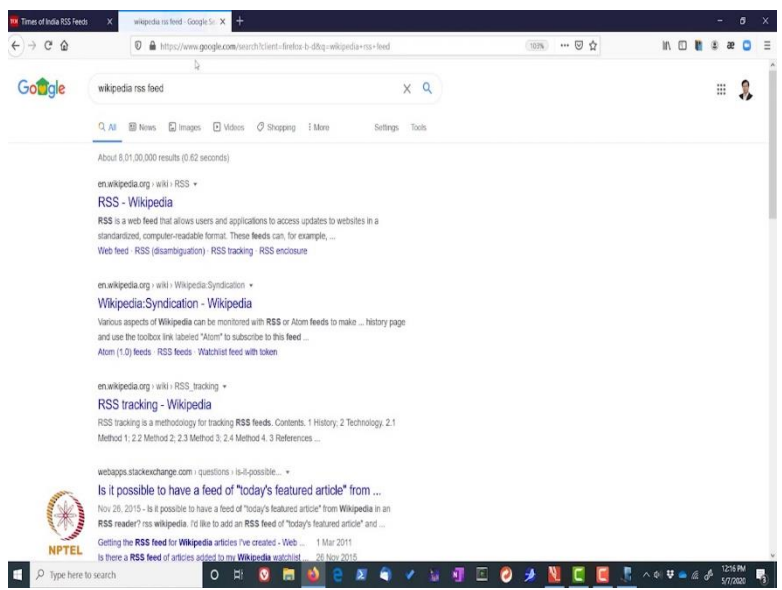
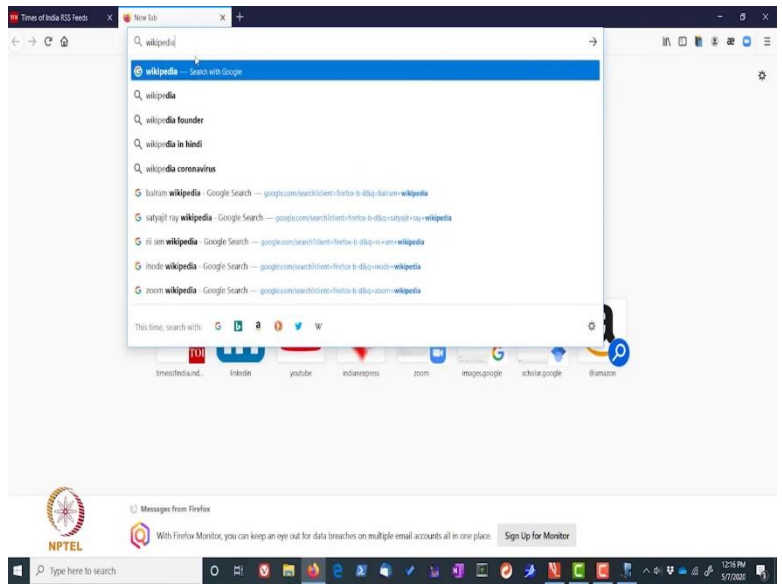
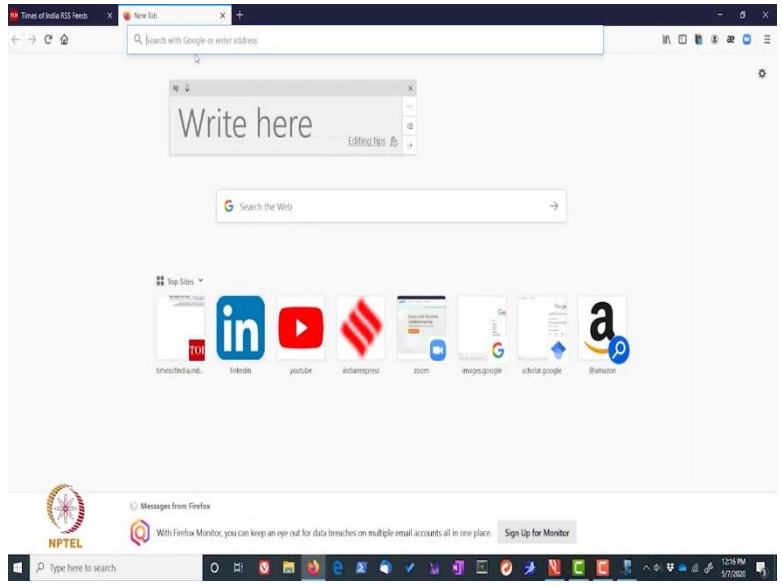
THE TIMES OF INDIA

Briefs TOI+ #MasKIndia Videos City India World Business Tech Cricket Sports Entertainment TV Web Series Lifestyle Blogs All Sections

Karipur
 Kishuta
 Ladhiana
 Mangalore
 Mysore
 Noida
 Pune
 Goa
 Chandigarh
 Lucknow
 Patna
 Jaipur
 Nagpur
 Rajkot
 Ranchi
 Surat
 Vadodra
 Varanasi
 Thane
 Thiruvananthapuram

World

US	3,100	NY YouKnow	Windows Live	neungattar	More
NRI	3,100	NY YouKnow	Windows Live	neungattar	More
Pakistan	3,100	NY YouKnow	Windows Live	neungattar	More
South Asia	3,100	NY YouKnow	Windows Live	neungattar	More
UK	3,100	NY YouKnow	Windows Live	neungattar	More
Europe	3,100	NY YouKnow	Windows Live	neungattar	More
China	3,100	NY YouKnow	Windows Live	neungattar	More



Times of India RSS Feeds x RSS - Wikipedia

https://en.wikipedia.org/wiki/RSS

Article Talk

Read View source View history Search Wikipedia

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Page history

RSS

From Wikipedia, the free encyclopedia


For other uses, see **RSS (disambiguation)**.
For RSS feeds from Wikipedia, see **Wikipedia:Syndication**.

RSS (originally **RDF Site Summary**; later, two competing approaches emerged, which used the backronyms **Rich Site Summary** and **Really Simple Syndication** respectively^[a]) is a web feed^[b] that allows users and applications to access updates to websites in a standardized, computer-readable format. These feeds can, for example, allow a user to keep track of many different websites in a single news aggregator. The news aggregator will automatically check the RSS feed for new content, allowing the list to be automatically passed from website to website or from website to user. This passing of content is called **web syndication**. Websites usually use RSS feeds to publish frequently updated information, such as blog entries, news headlines, or episodes of audio and video series. RSS is also used to distribute podcasts. An RSS document (called "feed", "web feed",^[c] or "channel") includes full or summarized text, and metadata, like publishing date and author's name.

A standard XML file format ensures compatibility with many different machines/programs. RSS feeds also benefit users who want to receive timely updates from favourite websites or to aggregate data from many sites. Subscribing to a website RSS removes the need for the user to manually check the website for new content. Instead, their browser constantly monitors the site and informs the user of any updates. The browser can also be commanded to automatically download the new data for the user.

RSS feed data is presented to users using software called a news aggregator. This aggregator can be built into a website, installed on a desktop computer, or installed on a mobile device.

RSS – Rich Site Summary

	
Filename extension	.rss, .xml
Internet extension	application/rss+xml (registration not finished) ^[1]
media type	Web syndication
Type of format	Web syndication
Extended from	XML

Type here to search

Permanent link
Page information
Wikidata item
Cite this page

In other projects
Wikimedia Commons

Print/export
Download as PDF
Printable version

Languages
العربية
বাংলা
Español
فارسی
हिन्दी
Português
Pycckий
සිංහල
ไทย
中文
Edit links

RSS feed data is presented to users using software called a news aggregator. This aggregator can be built into a website, installed on a desktop computer, or installed on a mobile device. Users subscribe to feeds either by entering a feed's URI into the reader or by clicking on the browser's feed icon. The RSS reader checks the user's feeds regularly for new information and can automatically download it, if that function is enabled. The reader also provides a user interface.

Contents [hide]

- 1 **History**
- 2 **Example**
- 3 Variants
- 4 Modules
- 5 Interoperability
- 6 Podcasting and RSS
- 7 BitTorrent and RSS
- 8 RSS to email
- 9 RSS compared with Atom
- 10 Current usage
- 11 See also
- 12 Notes
- 13 References
- 14 External links

History

This section needs to be updated. Please update this article to reflect recent events or newly available information. (October 2013)

Main article: *History of web syndication technology*

The RSS formats were preceded by several attempts at web syndication that did not achieve widespread popularity. The basic idea of restructuring information about websites goes back to as early as 1995, when Ramanathan V. Guha and others in AOL's *Advanced Technology Group* developed the *Meta Content Framework*.^[1]

Type here to search

My.Netscape.Com portal^[2] This version became known as RSS 0.9.^[7] In July 1999, Dan Libby of Netscape produced a new version, RSS 0.91,^[8] which simplified the format by removing RDF elements and incorporating elements from Dave Winer's news syndication format.^[9] Libby also renamed the format from RDF to **RSS Rich Site Summary** and outlined further development of the format in a "futures document".^[9]

This would be Netscape's last participation in RSS development for eight years. As RSS was being embraced by web publishers who wanted their feeds to be used on My.Netscape.Com and other early RSS portals, Netscape dropped RSS support from My.Netscape.Com in April 2001 during new owner AOL's restructuring of the company, also removing documentation and tools that supported the format.^[10]

Two parties emerged to fill the void, with neither Netscape's help nor approval: The RSS-DEV Working Group and Dave Winer, whose UserLand Software had published some of the first publishing tools outside Netscape that could read and write RSS.

Winer published a modified version of the RSS 0.91 specification on the UserLand website, covering how it was being used in his company's products, and claimed copyright to the document.^[11] A few months later, UserLand filed a U.S. trademark registration for RSS, but failed to respond to a USPTO trademark examiner's request and the request was rejected in December 2001.^[12]

The RSS-DEV Working Group, a project whose members included Guha and representatives of O'Reilly Media and Moreover, produced RSS 1.0 in December 2000.^[13] This new version, which reclaimed the name RDF Site Summary from RSS 0.9, reintroduced support for RDF and added XML namespaces support, adopting elements from standard metadata vocabularies such as Dublin Core.

In December 2000, Winer released RSS 0.92^[14] a minor set of changes aside from the introduction of the enclosure element, which permitted audio files to be carried in RSS feeds and helped spark podcasting. He also released drafts of RSS 0.93 and RSS 0.94 that were subsequently withdrawn.^[15]

In September 2002, Winer released a major new version of the format, RSS 2.0, that redubbed its initials Really Simple Syndication. RSS 2.0 removed the type attribute added in the RSS 0.94 draft and added support for namespaces. To preserve backward compatibility with RSS 0.92, namespace support applies only to other content included within an RSS 2.0 feed, not the RSS 2.0 elements themselves.^[16] (Although other standards such as Atom attempt to correct this limitation, RSS feeds are not aggregated with other content often enough to shift the popularity from RSS to other formats having full namespace support.)

Because neither Winer nor the RSS-DEV Working Group had Netscape's involvement, they could not make an official claim on the RSS name or format. This has fueled ongoing controversy^{[17][18]} in the syndication development community as to which entity was the proper publisher of RSS.

One product of that contentious debate was the creation of an alternative syndication format, Atom, that began in June 2003.^[17] The Atom syndication format, whose creation was in part motivated by a desire to get a clean start free of the issues surrounding RSS, has been adopted as IETF Proposed Standard RFC 4287.^[8]

In July 2003, Winer and UserLand Software assigned the copyright of the RSS 2.0 specification to Harvard's Berkman Klein Center for Internet & Society, where he had

Type here to search

Tones of India RSS Feeds x RSS - Wikipedia x

https://en.wikipedia.org/wiki/RSS

Featured content
 Current events
 Random article
 Donate to Wikipedia
 Wikipedia store

Interaction
 Help
 About Wikipedia
 Community portal
 Recent changes
 Contact page

Tools
 What links here
 Related changes
 Upload file
 Special pages
 Permanent link
 Page information
 Wikidata item
 Cite this page

In other projects
 Wikimedia Commons

Print/export
 Download as PDF
 Printable version

Languages
 Add all
 NPTEL


RSS (originally RDF Site Summary, later, two competing approaches emerged, which used the backronyms Rich Site Summary and Really Simple Syndication respectively^[d]) is a web feed^[d] that allows users and applications to access updates to websites in a standardized, computer-readable format. These feeds can, for example, allow a user to keep track of many different websites in a single news aggregator. The news aggregator will automatically check the RSS feed for new content, allowing the list to be automatically passed from website to website or from website to user. This passing of content is called web syndication. Websites usually use RSS feeds to publish frequently updated information, such as blog entries, news headlines, or episodes of audio and video series. RSS is also used to distribute podcasts. An RSS document (called "feed", "web feed",^[d] or "channel") includes full or summarized text, and metadata, like publishing date and author's name.

A standard XML file format ensures compatibility with many different machines/programs. RSS feeds also benefit users who want to receive timely updates from favourite websites or to aggregate data from many sites.

Subscribing to a website RSS removes the need for the user to manually check the website for new content. Instead, their browser constantly monitors the site and informs the user of any updates. The browser can also be commanded to automatically download the new data for the user.

RSS feed data is presented to users using software called a news aggregator. This aggregator can be built into a website, installed on a desktop computer, or installed on a mobile device. Users subscribe to feeds either by entering a feed's URI into the reader or by clicking on the browser's feed icon. The RSS reader checks the user's feeds regularly for new information and can automatically download it, if that function is enabled. The reader also provides a user interface.

RSS - Rich Site Summary



Filename extension	.rss, .xml
Internet media type	application/rss+xml (registration not finished ^[i])
Type of format	Web syndication
Extended from	XML

Contents [v] [d]

- 1 History
- 2 Example
- 3 Variants
- 4 Modules
- 5 Interoperability
- 6 Podcasting and RSS
- 7 Refinement and RSS

Type here to search

PDF Annotator

File Edit Tools View Options Window Help

main.pdf

Motivation

- The world wide web has a lot of information that changes frequently.
- There is no well defined publish-subscribe interface.
 - Publish : Post updates, and send to all the subscribers.
 - Subscribe : Register to get updates.
- Polling based methods are not efficient.
- Corona provides a scalable method to disseminate updates.

Motivation - II

- Content that changes frequently
 - Blogs, wikis, news sites
- Current solution - micronews syndication (RSS feeds)
 - Based on *active polling*.
 - Polling - tradeoff between latency at the client and bandwidth at the server.

NPTEL

Type here to search

Motivation Corona Design Evaluation Motivation Contemporary Approaches Related Work

Motivation

- The world wide web has a lot of information that changes frequently.
- There is no well defined publish-subscribe interface.
 - Publish : Post updates, and send to all the subscribers.
 - Subscribe : Register to get updates.
- Polling based methods are not efficient.
- Corona provides a scalable method to disseminate updates.

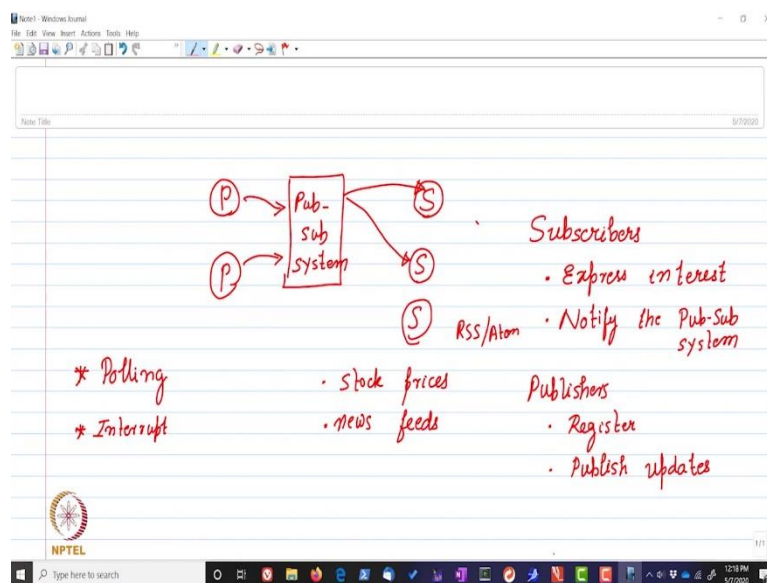
NPTEL

Smruti R. Sarangi Distributed Publish Subscribe Systems 4/34

So, what you see over here is a complicated, unreadable piece of text. But if I were to open the same in an RSS feed reader, then actually I would nicely see all of the updates in my feed reader, And So, a quick word about RSS if I were to show you the Wikipedia entry. So, RSS, the full form is a really simple syndication. And So, this gives us summaries of websites where it is one of the most popular Pub Sub systems, the websites, publish the news, and then we can subscribe to them.

And then I will immediately get notified the moment that there is a change in a website. So, Pub Sub systems are not just limited to RSS, even the RSS is very common. Pub Sub systems are kind of there everywhere, where we have a publisher that posts updates. And we have a subscriber that registers to get updates. So, we just saw a example of RSS. But of course, this can be in any kind of system. So, how does this actually work? So, the way that this actually works is kind of interesting. Let us go back to our journal over here.

(Refer Slide Time: 7:13)



So, the way that this works, when we can have 2 methods, the first is polling. So, in polling, what happens is that the subscribers keep on asking the publisher, has Do you have an update, do you have an update and the other is kind of an interrupt driven mechanism, where the subscriber registers her IP address with the Pub Sub system. And whenever there is a change, the subscriber is notified. So, clearly, in an interrupt driven system, the Pub Sub system has to do the work it has to maintain state, whereas in a polling-based system, the Pub Sub system does not maintain any state.

(Refer Slide Time: 8:11)

Motivation

- The world wide web has a lot of information that changes frequently.
- There is no well defined publish-subscribe interface.
 - Publish : Post updates, and send to all the subscribers.
 - Subscribe : Register to get updates.
- Polling based methods are not efficient.
- Corona provides a scalable method to disseminate updates.

Site Polling Site Pull

Polling Interrupt

NPTEL Srnul R. Sarangi Distributed Publish Subscribe Systems 4/34

So, the specific motivation for Corona other specific motivation for Corona is like this, that a polling-based methods are not very efficient, because they place a lot of load on the server. So, Corona, provides a very scalable method and very flexible method also to disseminate updates. So, let us say for example, I am interested in updates, sports updates. And let us say there are a million people like me, So, all millions of us should not actually be pulling the sports site, and actually trying to get updates. And also, so let me draw this over here. So, let us say there is a sports site, a website, and there are millions of subscribers.

So, we have discussed 2 methods polling and interrupt based. So, the polling-based method, the millions of subscribers go to the site and keep seeing if there is so, me new news or that there is so, me change. And clearly this increases the load on the site. So, most cricket lovers would actually go to the site cricket info, whenever a match is going on, and we cannot actually see the match. And pretty much we keep on pinging it to see if there is a new status update.

And the other is that actually the server essentially notifies the update every browser is maintained state. So, in that case, if the state changes, then each of the client browsers needs to be notified. So, this places this also places a large load on the server. So, the main aim of Corona is to place a degree of intermediaries over here, which minimize the load on the server, as well as stop the clients from polling. So, the intermediaries actually pull the server, but they minimize the load at the server.

And instead of a client's polling, they actually send messages to their clients whenever there is an update. So, the intermediaries maintain state. So, what is Corona once again, well, if there

is a very popular website, and there are a set of clients, it is a lot of clients, then Corona is kind of like a middle middleman sitting over here. Where each of the corona nodes maintain state. So, it keeps track of the clients. Whenever there is an update, it disseminates the updates the client, and the nodes themselves, keep polling the website, but they do it in such a way to minimize the load on the website.

So, the website is happy, the intermediate layer, which is Corona that we need to study. And of course, the clients are happy, because they quickly get the updates without, increasing the load on the server, because otherwise the server will crash. So, this is the basic idea the core idea of having a kind of a middle layer, So, me kind of an intermediate layer in a traditional Pub Sub system.

(Refer Slide Time: 11:35)

Motivation - II

- Content that changes frequently
 - Blogs, wikis, news sites (news, stock prices, sports (corca))
- Current solution – micronews syndication (RSS feeds)
 - Based on naive polling.
- Polling – tradeoff between latency at the client and bandwidth at the server.
minimize server load.
minimize latency of updates (for each client)

Smrutil R. Sarangi | Distributed Publish Subscribe Systems | 5/34

So, what is Pub Sub used for? Well, it is used for content that changes frequently. And for which we need to quickly know that the content has changed. So, these could be blogs, Wikipedia's new sites, but most common, it is like news, I would say that is the most common stock prices.

That is also very common sports scores, updates. Cricket, for example, all day, if a match goes on for the entire day, for the entire day, we are getting updates. So, the current Solution is using RSS feeds, as I just showed you on types of India's website is called micro news syndication. And it is kind of based So, now there are many more sites that also give us short news briefs. So, many of these sites like in shorts and So, on, give us very, short news briefs, but all of these are based on very simple polling.

So, there is clearly a tradeoff between the latency perceived at the client and the bandwidth at the server. So, both need to be taken care of. So, one is minimize the server loads the other is. latency of updates. The idea here is that the load on the server of course has to be minimized. But also, the latency of updates for each client even that also has to be reduced.

(Refer Slide Time: 13:42)

The slide shows an 'Outline' with the following structure:

- 1 Motivation
 - Motivation
 - Contemporary Approaches
 - Related Work
- 2 Corona Design
 - Design
 - Corona Schemes
 - Systems Management
- 3 Evaluation

The slide also features a navigation toolbar on the left, a top navigation bar with 'Motivation', 'Corona Design', and 'Evaluation', and a footer with 'Smruti R. Sarangi', 'Distributed Publish Subscribe Systems', and '6/34'.

The slide titled 'Current Solutions' contains the following bullet points:

- Content providers impose rate limits based on – IP addresses, range of IP addresses.
- Servers ask the client to stop polling, or to change their polling intervals.
- Corona's aim:
 - Manage the server's bandwidth efficiently.
 - Stay within the limits.
 - Give the clients the best possible update latency.

Handwritten in red ink is a diagram with the word 'akamai' in a central box. Three arrows point from this box to three separate boxes above it, and three arrows point from the box to three separate boxes below it, illustrating a distributed or multi-tier architecture.

The slide also features a navigation toolbar on the left, a top navigation bar with 'Motivation', 'Corona Design', and 'Evaluation', and a footer with 'Smruti R. Sarangi', 'Distributed Publish Subscribe Systems', and '7/34'.

So, what are the contemporary approaches? The contemporary approaches is that So, me content providers impose a rate limit on the amount of bandwidth per IP address the number of IP addresses that can connect at the same time or a range of IP addresses. So, service can also ask the client to stop polling or change their polling intervals. So, that is why many of the popular news sites like including types of India or anything else that has a large viewership,

So, actually they go via intermediaries. So, one of the most popular intermediaries is actually a provider called Akamai.

So, what Akamai does is that it essentially sits between some very popular websites and the clients. So, whenever a client browser actually points to some very popular website, it essentially hits an Akamai node. And his job was Akamai node to collect updates from these websites and give it to the client browsers. So, of course this process is hidden from us. So, as a client browser, we do not get to see it. But it is kind of still happening. That Akamai with sits in the middle that actually does it for us, which is that getting the updates latest updates from the server and giving them to the clients.

So, Corona's aim is something very similar. So, of course, Akamai is a proprietary solution, Corona is in academia. So, it manages the service bandwidth efficiently stays within limits, and gives the clients the best possible update latency. So, of course, for clients, you can think of this as something which can either be pulling or interrupt, but the relationship between the content aggregator or the provider like Akamai or Corona, and the website is a purely polling based relationship.

(Refer Slide Time: 15:48)

The slide is titled "Corona Front End" and contains the following content:

- Users subscribe by sending instant messages to a registered Corona userid. *gam*
- Corona → a cloud of nodes that monitors a set of channels
- A channel is a web page, or any other service that generates an active feed
- The Corona resource allocation algorithm dedicates a group of nodes to monitor each channel
- *channel* They filter out useless content – timestamps, advertisements
- *000* A feed specific difference engine extracts the relevant portions that have changed
- Distributes the changes to the clients *polling*
- difference
** distributed*

Navigation sidebar (left): Home, Back, Forward, Search, etc.

Footer: Smruti R. Sarangi | Distributed Publish Subscribe Systems | 8/34

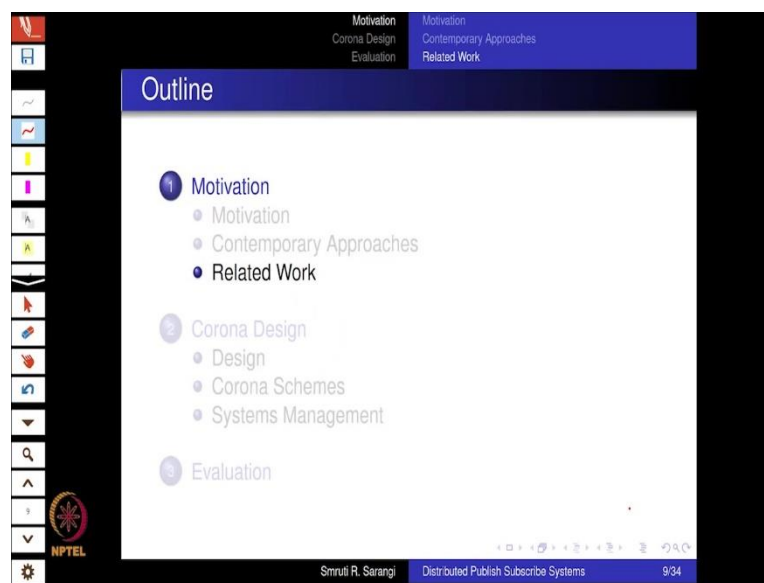
So, how did the corona system which was done way back in the 90s? How did that actually work? So, users subscribed by sending instant messages to a registered Corona user ID. So, of course, instant messages are there everywhere. So, we have a lot of instant messaging protocols like Google Talk, and so, on, like Skype and Google and we a lot of instant messaging platforms. Also, we have the Software called game in Linux, which kind of aggregates all of

them. So, they send an instant message to a Corona user ID. Corona is a set of nodes on the cloud, that monitor is set of channels.

So, essentially, every website or every content provider is called a channel. So, essentially, any service that generates an active feed an active feed of data, an active feed of news is called a channel. So, the corona resource allocation algorithm will essentially dedicate a set of nodes in Corona to monitor a given channel. So, they will do that. So, they will filter out useless content.

So, this can include timestamps and advertisements and things that actually clients do not need. And then a difference engine will extract the relevant portions that have changed. And these differences will be distributed, the new updates will be distributed to the clients. So, this is all that Corona does, which means it first does pulling it pulls the channel, it extracts the difference. And then it distributes the difference to their clients.

(Refer Slide Time: 17:43)



Motivation Corona Design Evaluation Motivation Contemporary Approaches Related Work

Background of Publish Subscribe Systems

- Publishers : Post content
- Subscribers : Subscribe to get relevant content
- Topic Based : (subset)
 - Publishers and subscribers are connected by a set of topics. Each topic is called a **channel**.
 - Subscribers get asynchronous updates for the channels.
- Content Based
 - X • Subscribers can make queries on the content, and receive results.
- Drawbacks of research prototype pub-sub systems: require custom interfaces, difficult to use
- Corona: backward compatible, easy to use (IM based)

NPTEL Smriti R. Sarangi Distributed Publish Subscribe Systems 10/34

So, no motivation. Now, we are kind of well-motivated. So, let us look at a few of the solutions and related work. So, the solutions and related work goal goes something like this. We have publishers who post the content, we have subscribers who subscribe to relevant content. And so, this process of subscribing to relevant content can be done in 2 ways. One can be topic based, which means that publishers and subscribers are essentially connected by a set of topics. And each topic is a channel. So, the topic for example, can be food. So, whenever there is any update on food, it is given to the clients.

So, this is not exactly how Corona works. So, Corona is content based, where subscribers can make queries on the well, let me take back my statement. So, Corona is actually channel based. And in this case, the channel is like a specific website or a specific provided, say topic, of course, can span across channels. In the sense multiple websites can be providing food. So, this is not exactly what Corona does is a single channel is kind of a subset of topic based. content based is again when we make Verizon the content. So, the content is not organized by topics but it is just like raw content.

And then we essentially asked Corona to go through all the content and find the relevant ones that are that need to be sent to the clients. So, content based is not something that Corona does. So, what it does is actually a subset of a topic-based Pub Sub, where we get updates from specific from a particular channel.

So, many of the research prototype Pub Sub systems require custom interfaces and they are difficult to use. So, one great advantage of Corona was that it is kind of compatible with all content providers. And it is rather easy to use it is based on instant messaging, instant

messaging is ubiquitous, it is available everywhere. And because of that the corona system is extremely popular.

(Refer Slide Time: 20:31)

Micronews Syndication

- Short updates of frequently changing data – news stories, blogs, social media posts
 - Typically use an XML based format (examples – RSS, Atom)
- Accessed via http over standard URLs *RSS feed readers*
- Use feed readers such as akregator to display data
- The server can inform the client when not to poll by using the cloud XML tag

NPTEL Smruti R. Sarangi Distributed Publish Subscribe Systems 11/34

So, we have already discussed RSS, what is it, it is short updates of frequently changing data. So, can be news stories, blogs, blog posts, So, they typically use an xml-based format to share very short updates. And we can access it via HTTP over standard URLs as we just saw, and there are dedicated RSS feed readers like aggregator. To display the data, so, then the server informs the client when to pull and when not to pull by using a certain tag called a cloud xml tag. So, the servers and RSS automatically or do some sort of rate control. And if let us say there is excessive polling excessive load at the server, the server's control the rate. But in the case of Corona, this is not required.

(Refer Slide Time: 21:37)

The slide displays an XML snippet for an RSS feed. The code is as follows:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<rss version="2.0">
  <channel>
    <title>My Home Page</title>
    <link>http://www.cse.iitd.ac.in/~srsarangi</link>
    <description>Homepage of S. R. Sarangi</description>
    <item>
      <title>Teaching</title>
      <link>teaching.html</link>
      <description>All teaching activities</description>
    </item>
    <item>
      <title>Research</title>
      <link>research.html </link>
      <description>Research Methodology </description>
    </item>
  </channel>
</rss>
```

The slide also features a navigation menu on the left with various icons, a title bar at the top with the text 'RSS Example', and a footer at the bottom with the text 'Smruti R. Sarangi Distributed Publish Subscribe Systems 12/34'.

So, in the example of an RSS, xml snippet, So, the first mentioned the version of xml, the version of text encoding the RSS version 2. So, this is a channel. So, the channel of course, can have a lot of things. So, in this case, let us say I updated my homepage. So, then, So, my homepage will come here along with a link that has been updated the description of my homepage, and the particular items that have been updated. So, let us say I updated my teaching page, and I updated my research page. So, both the updates, So, let us say just change the descriptions, for example.

So, both updates the update for the teaching, as well as the update for the research, both of them are showing up over here. also, for So, both of them that update for teaching and research. Show up are separate items in the RSS link in the RSS snippet. And a dedicated RSS feed reader displays these items to the reader.

(Refer Slide Time: 22:56)

Motivation Design
Corona Design Corona Schemes
Evaluation Systems Management

Outline

- 1 Motivation
 - Motivation
 - Contemporary Approaches
 - Related Work
- 2 Corona Design
 - Design
 - Corona Schemes
 - Systems Management
- 3 Evaluation

Smruti R. Sarangi Distributed Publish Subscribe Systems 13/34

Motivation Design
Corona Design Corona Schemes
Evaluation Systems Management

Corona - Cornell Online News Aggregator

- Key Features
 - Co-operative Polling - Assign multiple nodes to poll the same channel, and share updates.
 - Optimally distribute the task of polling
 - Corona poses this problem as an optimization problem, and solves it using the Honeycomb optimization toolkit.

Channel
clients
multiple nodes

Smruti R. Sarangi Distributed Publish Subscribe Systems 14/34

Now, that we come to the design of Corona So, Corona stands for Cornell online news aggregator. Because this was initially a project from the Cornell University, the key features of Corona are as follows. The first is cooperative polling, which means we assign multiple nodes to pull the same channel and share updates.

So, this is the same channel. And we are multiple nodes. Then they pull the same channel and they share the updates with a much wider base of clients. And of course, the number of nodes that we assigned to pull the same channel depends on the popularity of the channel, the nature of the content, the size of the content and how many clients are interested. So, this is posed as a global optimization problem. And the honeycomb optimization toolkit is used to actually solve it.

(Refer Slide Time: 24:20)

The slide is titled "Design of Corona" and contains the following text and diagrams:

- Corona uses a Pastry based overlay.
- Each channel has a unique channel identifier that is given a position along the Pastry ring.
- Corona defines a **wedge** around the channel that logically splits the set of nodes along the ring.
 - **Wedge** → A set of nodes sharing a common number of prefix digits with the channel's identifier.
- A channel has polling level l , if it is polled by all the nodes that have at least l matching prefix digits with it.
- A wedge associated with a channel polls for it.

Handwritten notes in red ink include: $b \rightarrow \text{base}$, $\text{URL} \xrightarrow{\text{SHA}} \text{hash}_b$, and $\text{IP} \rightarrow \text{hash}_b$. A diagram shows a circular ring with a wedge of size c and a label $\log_b N$.

Corona uses a pastry-based overlay. So, recall that in pastry, what we were doing is that we had created a large circular space, and we represented numbers in a given base. So, the first thing that we actually did so, consider a channel. So, for each channel, let us say a channel is uniquely identified by its URL from the URL using the SHA algorithm. We hashed it, it produced a hash value. So, the hash value was not exactly represented in binary.

But it was represented in base b . So, recall a small clarification over here, in a pastry paper actually the bases 2 the power b . But in the corona paper it is not 2 the power b , the convention is that the base is b . So, we will go with this convention, the corona convention, even though recall that in the pastry paper, the connotation of b was actually different and the actual base was to raise to the power b . But as I said, in Corona, we will go with the convention that the base in which the hash is being represented is not base b .

So, this hash is in base b . And for every such hash value, we can find a position for it in the array. So, the channel can be positioned over here. So, it is expected that, the match that will that it will have So, the nearest node So, the idea here also is the same that the closest node with the channel that is the owner of the channel and it is expected that the level of the match will be $\log_b N$ to the base b , So, that is the expected value. So, if it does not happen, of course, we will see what is done, but this is expected to be the common case.

So, what Corona actually does it or it defines a wedge around the channel, something like this, Something like a pizza slice. So, which means it logically splits the nodes on this. So, what it does is that for the circular ring, similar to pastry, it maps both the nodes as well as the channels,

the channels URL is hashed and mapped to the ring. And for the nodes, the IP address is hashed and mapped to the ring.

So, once all of the ones all the nodes and channels, find that place on the ring, we can define a wedge for a wedge is defined as a set of nodes that share a common number of prefix digits with the channels identified. Also, if a channel has polling level 1, this means that it is pulled by all the nodes that have at least 1 matching prefix digits with it.

(Refer Slide Time: 27:40)

The image consists of two screenshots of a Windows Journal window, showing handwritten notes and diagrams. The top screenshot features a circle divided into segments, with labels 'channels' and 'nodes' on the left. A 'wedge' is highlighted in a segment. To the right, there are handwritten notes: 'Optimization: wedge sizes', 'C → 0x A1234BCDEF', 'N → 0x A1234DEF...', 'l = 5', and 'log_b N'. Below the circle, it says 'l matching prefix digits with the channel id' and 'magic → all the nodes in a wedge poll the channel'. The bottom screenshot is identical but has a 'Pen and Highlighter Settings' dialog box open over the diagram.

So, let us explain this in Some more detail. Because this is by far the most important concept in Corona. So, both the channels as well as the nodes are mapped to the circle. So, let us say that the channel is mapped over here. So, then one of the nodes will be the closest to the channel. So, this node is made the owner of the channel. So, let us say this node over here, So,

this node is the owner of the channel. So, it is expected, the degree of the prefix match will be login to the base b . So, of course, if it does not happen, we still find the closest on it and we reduce the prefix match.

Now what we do is that around each channel we define a wedge So, every wedge So, wedge is like a pizza slice. So, with every wedge, we define a polling level l . So, the polling level l basically means that there is l matching. So, the URL matching prefix digits with the channel ID. So, let us say that the channel ID we are representing in base 16. So, let us say something like this. And let us assume by prefix we mean from left to right.

So, if this is the channel, then for a given node if let us say it is a part of the wedge that is monitoring the channel. So, I will come back to what is monitoring the channel. But let us look at this for the time being. And let us say diverges from here on. So, in this case, the polling level is equal to the number of matching prefix digits. In this case, the number of matching prefix digits is five. So, the polling level is five.

So, we can always define a wedge based on the polling level, lower is the polling level, the wider is the wedge, higher is the polling level, the narrower is the wedge. And of course, once the polling level process login to the base b , we expect the size of the wedge to be 0. So, at the beginning, the size of the wedge will just contain just the node and the channel.

And then the polling level is expected to be login to the base b . But of course, if the channel is very popular, we would like a lot of nodes to pull for the channel. So, what we actually do is we partition the hash space in pastry by defining a wide wedge, how do we do this by reducing the polling level, such that we have a wide wedge over here.

And the magic of Corona comes here, that all the nodes within this edge pool for the channel. So, what is the magic of Corona? Well, the magic of Corona the key, key, key idea is that all the nodes. Is that all the nodes in a wedge poll the channel. So, if you just want more nodes to poll the channel, well, all that we have to do is that we have to reduce the polling level. Once we reduce the polling level, we will have more nodes that it will span. And the more and the increased number of nodes will actually come and pull the channel. And let us say the channel loses popularity, then all that we need to do is that we need to increase the polling level, the wedge will shrink.

So, let us say the wedge might become something like this. And then a fewer number of nodes will poll the channel. So, on a similar line, we can have different channels or different parts of

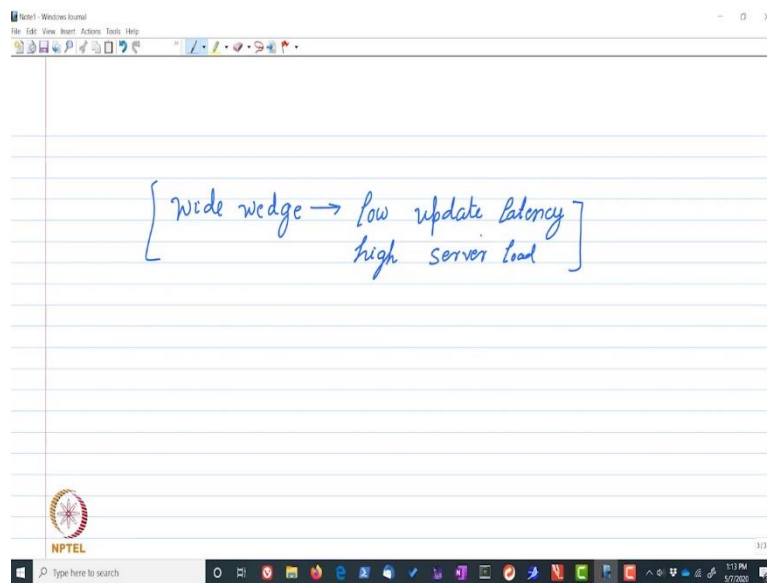
the (32:34) for each of them, we will have a wedge that is defined. Also note that in this case, wedges can be overlapping.

So, in this case, let us see if there is another channel over here, let us say something like this, we can have, let me use a different color. Let us see if I can do that in settings. I can. So, let us say the other channel over here, which is this channel, then essentially, I can define a wedge for it, which would look something like this.

So, here kindly note that the 2 wedges are overlapping. So, nodes in the overlapping wedges would actually pull for both. So, Corona does have a way of managing such conflicts in a very indirect manner. But we will discuss that later. As long as the key concept is clear. And the key concept over here is the concept of wedges, which are essentially partitions of the hash space. So, the concept of wedges is the most important. And it is essentially a partition of a hash space. And that is clearly the most important concept here that we can either have a wide wedge or a narrow wedge, depending on the polling level, which can be changed dynamically.

So, the entire optimization problem within Corona in Corona the entire optimization problem is to essentially figure out the wedge sizes for each other. So, that would talk about the server load that would talk about the load on the corona nodes. And that will also determine the update latency of the client. So, all of this related to the web size, well how So, if the web size if the wages are very wide, this will mean that we are getting a lot of updates from the server the update latency will be low.

(Refer Slide Time: 34:42)



So, I can maybe summarize this over here. Let us say a wide wedge. Low update latency because a lot of nodes are pulling, and they can also pull at random intervals. So, with that, what will happen is that, if there is any change, it will quickly get caught by one of the nodes in the wedge, it can then distribute the updates to the rest of the nodes.

So, wide wedge will have a low update latency and a high server load and vice versa narrow wedge will have a high update latency and a low server load. So, we need to find this and we need to So, there are conflicting requirements of the update latency and server loads, we will see how Corona solves it.

(Refer Slide Time: 35:40)

A slide titled "Design of Corona" from a presentation. The slide has a blue header with navigation tabs: "Motivation", "Corona Design", "Evaluation", "Design", "Corona Schemes", and "Systems Management". The main content area is white with a blue border. It contains a list of bullet points and a diagram. The diagram shows a circular ring with a wedge-shaped segment labeled 'C'. To the right of the ring is the text $\log_2 N$. Above the ring, there is a flow diagram: $z \rightarrow \text{ID}$ with an arrow pointing to $\text{URL} \xrightarrow{\text{SHA}} \text{hash}$ and $\text{IP} \rightarrow \text{hash}$. The bullet points are:

- Corona uses a Pastry based overlay.
- Each channel has a unique channel identifier that is given a position along the Pastry ring.
- Corona defines a **wedge** around the channel that logically splits the set of nodes along the ring.
 - **Wedge** \rightarrow A set of nodes sharing a common number of prefix digits with the channel's identifier.
- A channel has polling level l , if it is polled by all the nodes that have at least l matching prefix digits with it.
- A wedge associated with a channel polls for it.

The footer of the slide includes the NPTEL logo, the name "Smruti R. Sarangi", the text "Distributed Publish Subscribe Systems", and the slide number "15/34".

So, we understood the notion of a wedge the common number of prefix digits, a channel has falling level l , if it is polled by all the nodes with l matching prefix digits with it, which means all the nodes within the wedge.

(Refer Slide Time: 35:58)

Avg. Detection Time and Load on the Server

- Let the total number of nodes be N .
- A channel with polling level l , has on an average N/b^l nodes in its wedge
- Let τ be the polling interval.
 - Average detection time for updates = $\frac{\tau}{b^l} \cdot \frac{N}{b^l}$
 - Collective load placed on the server $\propto \frac{\tau}{b^l} \cdot \frac{N}{b^l}$

Problem
 Problem: Estimate the polling levels of each channel, or, alternatively, the sizes of the wedges.

So, a little bit of math over here, let the total number of nodes be N , a channel with polling level l will have on an average N/b^l nodes in its wedge. So, I am not proving this instead, I am directing the viewers to the pastry paper that proves this in great detail. So, this paper has to be read as a prerequisite before looking at before leaving this lecture. So, we will assume that this is the this is kind of a gospel truth here. It should be τ . So, let tau be the polling interval. So, the average detection time for updates, so, the average detection time. So, if let us say we call it that, so, let us give it this name, det time.

So, the det time is clearly proportional to the reciprocal of this are inversely proportional to the number of nodes in a wedge higher or the number of nodes in the wedge. So, assuming that they are pulling at random intervals, which is the case, the detection time will be inversely proportional to it. So, we can see it is $\frac{b^l}{N}$. Furthermore, the detection time will also be proportional to the average polling interval, which means that for every server if it is polling, once every tau seconds detection time will be proportional to that.

And in addition, given the uniform distribution that you are assuming here, which means that let us say that if a server is polling, and between time 0 and τ , it could have been changed, the data could have changed at any point in time, the average latency will be $\frac{\tau}{2}$ which is easy to

prove. So, this gives us the average detection time for updates to be $\frac{\tau}{2} \cdot \frac{b^l}{N}$. And the collective node placed on the server is clearly proportional to the size of the wedge, which is $\frac{N}{2^l}$. So, the problem is to estimate the polling levels of each channel or alternatively, the sizes of the wedges.

(Refer Slide Time: 38:17)

Motivation Corona Design Evaluation Design Corona Schemes Systems Management

Outline

- Motivation
 - Motivation
 - Contemporary Approaches
 - Related Work
- Corona Design
 - Design
 - Corona Schemes
 - Systems Management
- Evaluation

Smruti R. Sarangi Distributed Publish Subscribe Systems 17/34

Motivation Corona Design Evaluation Design Corona Schemes Systems Management

Corona-Lite

- Ensures good update performance, while ensuring that the load on the servers is light.

Optimization Problem

minimize $(\sum_1^M q_i \frac{b^l}{N})$ such that $(\sum_1^M s_i \frac{N}{b^l} \leq \sum_1^M q_i)$

- M Number of channels
- q_i Number of clients for channel i # clients
- l_i Polling level of channel i
- N Number of nodes
- s_i Content size for channel i

bandwidth *justifies CORONA* $q_i \frac{b^l}{N}$

Smruti R. Sarangi Distributed Publish Subscribe Systems 18/34

Motivation Corona Design Evaluation Design Corona Schemes Systems Management

Avg. Detection Time and Load on the Server

- Let the total number of nodes be N .
- A channel with polling level l , has on an average N/b^l nodes in its wedge
- Let τ be the polling interval.
 - Average detection time for updates = $\frac{\tau}{2} \frac{N/b^l}{N}$
 - Collective load placed on the server $\propto \frac{N/b^l}{N} \tau$

Problem
 Problem: Estimate the polling levels of each channel, or, alternatively, the sizes of the wedges.

NPTEL Smruti R. Sarangi Distributed Publish Subscribe Systems 16/34

So, let us now discuss a few of the corona schemes. So, the first is Corona lite, which is also the first game that comes to our mind when we think about this problem. Intuitively, this ensures good update performance, while ensuring that the load on the servers is light. So, let us first look at the terminology.

Let M be the number of channels. q_i be the number of clients for channel i , write is important q is for number of clients. For channel i . l_i is a polling level of channel i , N is the total number of nodes. And s_i is a content size for channel i of course, appropriately normalized this is important. And this is not something that the paper stresses on was important for me to stress that it has to be appropriately normalized.

So, it is the normalized content size for channel i . So, what are we trying to do here? Well, what we are trying to do is that we are trying to $\sum_l^M q_i \frac{b^{l_i}}{N}$. So, let us see what this is. So, $\frac{b^{l_i}}{N}$. Just let us just go to the previous slide. So, this is proportional to the update time the update detection time. It is proportional to this quantity. So, we say that this is the update detection time. And this we are kind of waiting with a number of clients for channel i .

And some moment, we want to minimize this quantity, which is the weighted sum of the update detection time. Let me repeat that this quantity is the weighted sum of the update detection time where the weight is the number of clients for channel i . We want to minimize that. So, what effect will this have? Well, the effect that this will have is that for any channel that has a large number of clients, we would like to minimize the update time as much as possible.

So, essentially, for popular channels, we want to deliver updates as quickly as possible. And we essentially want to penalize less popular channels. So, this is of course, objective function, what is the constraint? The constraint is, so, $\frac{N}{b_i}$, which is the same as this quantity over here is essentially the expected size of the wedge. So, this is the expected size of the wedge. And this is the size of the content of the channel. So, if I multiply the content with the size of the wedge, this gives me an idea of the bandwidth requirement from the server to kind of supply data to all the corona nodes.

So, this is an estimate of the bandwidth of the server. So, if I were to add up the collective bandwidth are all the servers for all the channels. So, the collective bandwidth for all the channels has to be less than equal to the number of clients. And this is like kind of a bandwidth constraint. The reason that there is a bandwidth constrained is that, look, if this was equal to the number of clients, we did not need Corona. So, one main advantage of Corona is that we want to pretty much reduce the bandwidth at the bandwidth requirement at the server.

Because otherwise, the server could just provide it provide data to all the clients, we did not need Corona in the first place. So, we keep this as a constraint that the normalized content size multiplied, of course, with the size of the wedges, the sum of that across the channels should be less than equal to the total number of clients for all the channels.

Switch, if you see from a common-sense point of view also makes sense, because this kind of justifies Corona. If we did not have this, then pretty much, Corona, what would happen is that in the quest of minimizing the update time, the bandwidth requirement from all the servers would be more than what a non-Corona system would require, which will not make any sense at all. So, given this kind of a common-sense constraint, we tried to minimize the update time.

(Refer Slide Time: 43:09)

The slide is titled "Corona-Lite - II" and is part of a presentation on "Distributed Publish Subscribe Systems". The navigation menu includes "Motivation", "Corona Design", "Evaluation", "Design", "Corona Schemes", and "Systems Management". The slide content consists of three bullet points:

- Clients of popular channels gain a lot, because the average update time gets reduced.
- Nicely partitions bandwidth across channels.
- Update performance would vary depending on the type of the workload.

At the bottom of the slide, the presenter's name "Smruti R. Sarangi" and the time "19:34" are visible.

So, here, what happens is the clients are popular channels, needless to say, gain a lot. Because for them, the average update time gets reduced that to significantly. This nicely partitions the bandwidth across the channels. The update performance, of course, would vary depending upon the type of the workload because the type of the workload is not being considered here, very explicitly. And for less popular channels, they kind of suffer in this case.

(Refer Slide Time: 43:43)

The slide is titled "Corona-Fast" and is part of a presentation on "Distributed Publish Subscribe Systems". The navigation menu is the same as in the previous slide. The slide content includes a mathematical optimization problem and a list of variables:

minimize $(\sum_1^M s_i \frac{N}{b_i})$ such that $(\sum_1^M q_i \frac{b_i}{N} \leq T \sum_1^M q_i)$

Handwritten notes in red ink include "weighted update time" pointing to the constraint, "fasted" pointing to the objective function, and a circled expression $s_i \frac{N}{b_i}$.

Variables defined:

- M Number of channels
- q_i Number of clients for channel i
- l_i Polling level of channel i
- N Number of nodes
- s_i Content size for channel i
- T Performance target

Aim
Minimize the load placed on the content servers, and achieve a target update time

At the bottom of the slide, the presenter's name "Smruti R. Sarangi" and the time "20:34" are visible.

Motivation Corona Design Evaluation Design Corona Schemes Systems Management

Corona-Lite

- Ensures good update performance, while ensuring that the load on the servers is light.

Optimization Problem

minimize $\left(\sum_1^M q_i \frac{b_i}{N}\right)$ such that $\left(\sum_1^M s_i \frac{N}{b_i} \leq \sum_1^M q_i\right)$

M Number of channels
 q_i Number of clients for channel i # clients CORONA
 l_i Polling level of channel i
 N Number of nodes
 s_i Content size for channel i

Handwritten annotations: "bandwidth" points to $\frac{N}{b_i}$, "justify" points to the constraint, and $q_i \frac{b_i}{N}$ is written in red.

NPTEL Smruti R. Sarangi Distributed Publish Subscribe Systems 18/34

So, then we define Corona fast, which is expected to be the fastest. So, I will explain in a second, why the terminology remains the same for M is the number of channels. q_i is the number of clients for Channel i , l_i is the polling level of channel i . N is the total number of nodes. s_i is the content size, and T is a certain constant, it is a performance target. So, what am I trying to do over here I am trying to minimize this quantity, what is this quantity? So, this this quantity is exactly the same as this quantity. In this case, s_i is the content size, as I said normalized, multiplied by the size of the image.

So, in this case, what I am trying to do is I am trying to minimize the bandwidth, the load placed on the content servers and trying to minimize that subject 2. So, this is interesting, this quantity is the same as this for Corona light, which is the. This is the weighted update time. So, this is essentially the update time weighted by the number of clients. So, we want to keep the update time less than a certain threshold. So, there are 2 aspects to this threshold, l_i is a constant, and the other is the total number of clients.

So, of course, the update time should be a function of the total number of clients. Because as the number of clients increase, what that would essentially do is that that would place a higher load. And here what we are saying is that, look, if the total number of clients are increasing, then the system is expected to be less responsive, the update time should increase. And that is kind of getting captured over here. And T is just a factor for normalization. But the key aim over here is that look, I want to achieve a target update time and the target update time is should be less than a threshold.

And the threshold is proportional to the total number of clients across all the channels, which also it should be that is a less clients, I should expect a lower update time. And if there are a lot of clients, I should expect a larger update. So, Corona fast is clearly the fastest when it comes to update time. The question is why? So, an astute reader can always ask that look for Corona lite, you try to explicitly minimize the update time. Whereas in Corona fast, you try to minimize the bandwidth. So, from a common-sense point of view, Corona lite should give the fastest update time when Corona fast should not. Well, that is true. But that is not the case.

In this case, we are like kind of forcing the update time to be less than a certain value. So, this is like a knob, which is totally under our control. And we are turning the knob to a point where update time is the lowest possible time that is possible. And in this case, the bandwidth is kind of going free. So, we are trying to minimize it. But think of the update time has been constrained and the bandwidth has been free.

So, we can lower this as much as we can, you know to achieve kind of any target within a feasible range. And once we have kind of set up our mind on this, then we can of course, So, once we set up this threshold over here, which is the constant T , which is the most important that sets our target.

Once the target has been set, we minimize the bandwidth. So, clearly, this is totally in our hands, we can make corona as fast as possible. Whereas in this case, the target is the bandwidth. And then of course, space for reducing the update time is kind of constrained. That is why Corona fast, as the name suggests, is genuinely fasted in terms of update time, as compared to Corona lite. I would ask the viewers to go over this logic several times till they are convinced.

(Refer Slide Time: 48:25)

The screenshot shows a presentation slide titled "Corona-Fast" with a blue header. The slide content includes a list of bullet points and a red-bordered box titled "Negative Aspects". Handwritten red annotations include the word "Minimize" above the first bullet point, an arrow pointing to "Bounds", and a bracket around the "Negative Aspects" box.

Corona-Fast

- ^{Minimize} Bounds the total amount of network traffic.
- Allows us to tune the update performance per application. For example, a stock market application might choose a very fast update performance.
- Along with providing applications the desired level of update performance, it can shield web servers from spikes in network load.

Negative Aspects

- Both Corona-Lite and Corona-Fast do not consider the rate of change of objects in the channel.
- Corona-Fair takes this into account

NPTEL Smruti R. Sarangi Distributed Publish Subscribe Systems 21/34

So, this of course bounds a total amount of network traffic that it indeed it does. So, well it does not bound, if you would actually see it actually tries to minimize I should maybe change this it allows us to tune the update performance per application, which is just what I said, given an application we can fix a target. For example, for a stock market application, we might choose a very fast update performance.

So, along with providing application, the desired level update performance, it tries to shield web servers or spikes in load reducing the band. So, of course, the negative aspects for both of these protocols is that both Corona lite as well as Corona fast. Do not consider the rate of change of objects in the channel. In the sensor load the update time only makes sense in the context of the rate of change of the object. So, the object is not changing. There is no point in actually polling the channel. So, Corona Fair takes this into account.

(Refer Slide Time: 49:41)

Motivation: Corona Design, Evaluation; Design: Corona Schemes, Systems Management

Corona-Fair

minimize $\left(\sum_1^M q_i \frac{\tau b^i}{N}\right)$ such that $\left(\sum_1^M s_i \frac{N}{b^i} \leq \sum_1^M q_i\right)$

- M Number of channels
- q_i Number of clients for channel i
- l_i Polling level of channel i
- N Number of nodes
- s_i Content size for channel i
- T Performance target
- u_i Update interval for channel i
- τ Polling interval

Handwritten notes:
 update interval $\rightarrow u_i$
 update time (perceived by the client)

NPTEL | Smruti R. Sarangi | Distributed Publish Subscribe Systems | 22/34

Motivation: Corona Design, Evaluation; Design: Corona Schemes, Systems Management

Corona-Lite

- Ensures good update performance, while ensuring that the load on the servers is light.

Optimization Problem

minimize $\left(\sum_1^M q_i \frac{b^i}{N}\right)$ such that $\left(\sum_1^M s_i \frac{N}{b^i} \leq \sum_1^M q_i\right)$

- M Number of channels
- q_i Number of clients for channel i
- l_i Polling level of channel i
- N Number of nodes
- s_i Content size for channel i

Handwritten notes:
 bandwidth
 justifies corona
 # clients
 $q_i \frac{b^i}{N}$

NPTEL | Smruti R. Sarangi | Distributed Publish Subscribe Systems | 18/34

Motivation: Corona Design, Evaluation; Design: Corona Schemes, Systems Management

Corona-Fair Sqrt and Log

Corona-Fair Sqrt

minimize $\left(\sum_1^M q_i \frac{\sqrt{\tau} b^i}{N}\right)$ such that $\left(\sum_1^M s_i \frac{N}{b^i} \leq \sum_1^M q_i\right)$

Corona-Fair Log

minimize $\left(\sum_1^M q_i \frac{\log(\tau) b^i}{\log(u_i) N}\right)$ such that $\left(\sum_1^M s_i \frac{N}{b^i} \leq \sum_1^M q_i\right)$

NPTEL | Smruti R. Sarangi | Distributed Publish Subscribe Systems | 23/34

So, it introduces a few additional variables. So, recall that this structure $q_i b$ and N is similar to Corona lite $q_i b$ and N . So, this is kind of Corona lite plus plus where this is the normalized update time $q_i b / N$ multiplied with τ which is the polling interval they should have been there and also the polling interval can be different for different channels.

This of course, this variant of Corona does not consider, but this is something which can be considered and the other is the update interval for channel i because for different channels, the update intervals will be different. For example, for stock prices, the update interval will be low, particularly when trading is happening, some stocks will be changing very quickly.

But once the stock market has closed for the day, the update interval will actually be very high right, because the stock prices are not going to change the stock prices will only change when the market reopens. So, any kind of a Corona polling should take this into account and the rate of change of the content is what is coming up over here.

So, let us say that the rate of change is very high, then u_i will be low. So, this constant will be high, if this constant will be high, then essentially, we want to minimize the update time. But let us say that we have a very, very slowly changing channel u_i will be high, this fraction will have a low value.

So, then of course, we can afford to have a high update time for the channel. And this is our regular bandwidth constraint, which we had in Corona lite, if I were to show you, So, is the same constant that we had over here, which is what we have over here. So, this remains the same, the only addition is the update interval.

So, we are using 2 terms here, update interval and update time. So, update interval is a property of the channel and the channel only. This is u_i . And the update time is a time that a client perceives between the actual update to the channel happening and the client actually seeing it, that is the update time.

So, the update time recall the keep in mind is different concept and it is what is perceived by the client. So, of course, it can be one criticism of this formula is that for very very slowly changing channels to update time can be very large, we just kind of unfair. So, we are basically giving too much importance to u_i to kind of reduce the importance in u_i .

What some, so, 2 other variants of Corona what they do is fair square root actually consider the square root of this quantity to kind of temper down the effects of u_i . And if you further

want to temper down the effects of u_i , instead of the square root we consider the log. So, that will further kind of reduce the effects of a large or small u_i . And So, we consider the ratios of the log.

(Refer Slide Time: 53:30)

The slide is titled "Corona-Fair" and is part of a presentation on "Corona Design". It contains three bullet points:

- Similar to Corona-Lite → minimizes update detection time, with a limit on the total amount of traffic
- Introduces a term to reduce the number of allocated servers if the rate of updates is small.
- It is possible to dampen this term by considering the square root or the log.

A red arrow points to the word "dampen" in the third bullet point.

The slide is titled "Corona-Fair Sqrt and Log" and compares two optimization problems:

Corona-Fair Sqrt
 minimize $\left(\sum_1^M q_i \frac{\sqrt{\tau} b_i}{\sqrt{u_i} N} \right)$ such that $\left(\sum_1^M s_i \frac{N}{b_i} \leq \sum_1^M q_i \right)$

Corona-Fair Log
 minimize $\left(\sum_1^M q_i \frac{\log(\tau) b_i}{\log(u_i) N} \right)$ such that $\left(\sum_1^M s_i \frac{N}{b_i} \leq \sum_1^M q_i \right)$

A red arrow points to the square root term in the Corona-Fair Sqrt equation.

So, as we discussed, Corona fair is similar to Corona Lite Corona fair, of course has 2 versions where square root and corona log. It also tries to minimize the update detection time, which we call the update time in the lecture. So, the Call Update time is different from Update Interval, just go back to 2 slides back.

So, it minimizes the weighted update detection time with a limit on the total amount of traffic. It also introduces a term to reduce the number of allocated servers, the date of updates is small, which we have been calling as the update interval. Furthermore, it is possible to dampen this

term damping by considering the square root the ratios of the square roots are the logs as we just saw over here.

(Refer Slide Time: 54:19)

Motivation Design
Corona Design Corona Schemes
Evaluation Systems Management

Decentralized Optimization

- Core optimization problem:
$$\min. \sum_{i=1}^M f_i(l_i) \text{ s.t. } \sum_{i=1}^M g_i(l_i) \leq T$$
- f_i and g_i are the performance or the bandwidth cost of the channel at polling level l_i .
- The values of l_i are integers.
- This problem is NP-Hard (need to compute a fast approximation)
- Honeycomb finds a solution in $O(M \log M \log N)$ time, which is optimal for M channels.

NPTEL
Smruti R. Sarangi Distributed Publish Subscribe Systems 25/34

So, this can be posed as a core optimization problem. So, the optimization problem, in this case will be something like this, that we want to minimize the quantity f subject to some other quantity being lower than the threshold. So, these can be kind of expressed as generic functions as f and g where we want to minimize f subject to G being lower than a threshold. If f_i and g_i are the performance, or the bandwidth cost of the channel at polling level l_i , so, it does not matter what these are, but, for different variants of Corona, f and g will be different functions, then we can have, you can set up an optimization problem.

And if you would see the corona paper, it proves to the problem is NP hard. But, of course, we can find good approximations to it, where m being the number of channels N being the number of nodes, we can find good approximations within $O(M \log M \log N)$ time. Honeycomb can do that.

(Refer Slide Time: 55:32)

Motivation Design
Corona Design Corona Schemes
Evaluation Systems Management

Decentralized Optimization

- Honeycomb combines channels with similar tradeoffs into a tradeoff cluster.
- Honeycomb nodes periodically exchange these clusters.
- Periodically, nodes run the Honeycomb algorithm to figure out the assignment of nodes to channels.
- Disagreement regarding the assignment of nodes to channels can be a problem !!!

distributed mechanism

NPTEL Smruti R. Sarangi Distributed Publish Subscribe Systems 24/34

Motivation Design
Corona Design Corona Schemes
Evaluation Systems Management

Decentralized Optimization

- Core optimization problem:

$$\min. \sum_1^M f_i(l_i) \text{ s.t. } \sum_1^M g_i(l_i) \leq T$$

- f_i and g_i are the performance or the bandwidth cost of the channel at polling level l_i .
- The values of l_i are integers.
- This problem is NP-Hard (need to compute a fast approximation)
- Honeycomb finds a solution in $O(M \log M \log N)$ time, which is optimal for M channels.

NPTEL Smruti R. Sarangi Distributed Publish Subscribe Systems 25/34

So, how does honeycomb actually do it? Well, the honeycomb, what it does is that it combines channels with similar tradeoffs, where the tradeoffs are defined as the fng functions into a tradeoff cluster. channels with similar behavior are nodes and channels, similar behavior are clustered. And the nodes periodically exchanged these clusters. And furthermore, nodes locally run the honeycomb algorithm to figure out the assignment of nodes to channels. And of course, disagreement regarding the assignment of nodes to channels can be a problem with this, of course, does not happen.

But the key idea over here is that there is a distributed mechanism of solving the optimization problem. So, it is not that one central site solves it. Instead, what happens is that if let us say this is the wedge, each of the nodes individually solves the problem. And each of the nodes

individually can on its own volition increase or reduce the polling level. For example, let us say if this node perceives that it needs to reduce its polling level then what can what it can do is that it can reduce the polling level on volition and consider a bigger part of pastry. bigger part of the pastry ring.

(Refer Slide Time: 57:14)

Motivation Corona Design Evaluation Design Corona Schemes Systems Management

Outline

- 1 Motivation
 - Motivation
 - Contemporary Approaches
 - Related Work
- 2 Corona Design
 - Design
 - Corona Schemes
 - Systems Management
- 3 Evaluation

Smruti R. Sarangi Distributed Publish Subscribe Systems 27/34

Motivation Corona Design Evaluation Design Corona Schemes Systems Management

System Management

- Each channel in Corona hashes its content to get its unique Pastry key.
- It is assigned an owner node in Pastry.
- For added fault tolerance, a key is assigned to F succeeding nodes.
- Owners receive subscriptions, and send updates to the all the subscribers.
- Corona manages co-operative polling through three mechanisms:
 - **Optimization Phase** : Applies Honeycomb based optimization to the traffic data collected from servers.
 - **Maintenance Phase** : Changes to polling levels are communicated to peers.
 - **Aggregation Phase** : Nodes receive tradeoff data from other peers.

Smruti R. Sarangi Distributed Publish Subscribe Systems 28/34

Now, some other systems management issues. So, let us just go over the entire process of running the corona system. So, each channel in Corona hashes its contents to catch this unique pastry key this we have seen, it is assigned an owner node the same way that pastry assigns it. For added fault tolerance, a key is assigned to F succeeding nodes.

So, if you would recall, we had seen something very similar in Amazon dynamo, very key is not just assigned to one node. But it is actually assigned to a series of nodes F successive nodes.

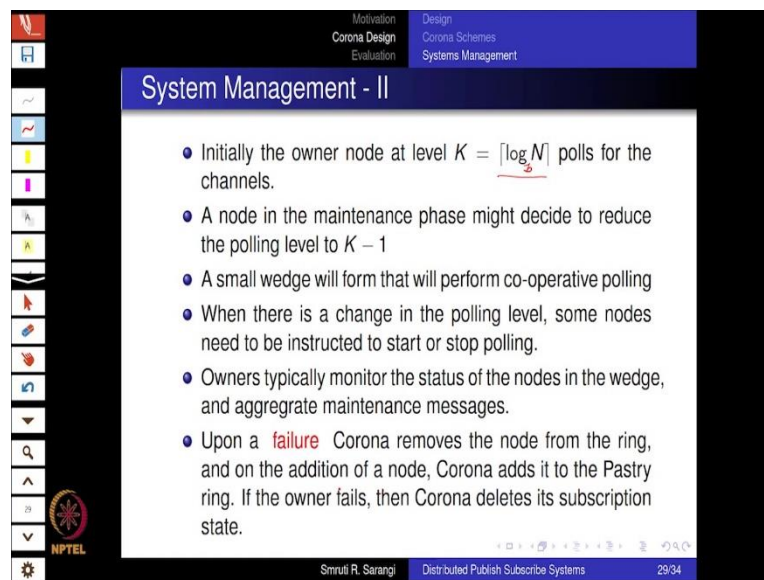
So, is that if one fails, the rest can take over. Owners receive subscriptions and send updates to all the subscribers.

So, what are the owners do owners are the one who receive all the subscriptions, and then they send updates to all the subscribers, subscribers typically. And also, we have cooperative polling, which means that all the nodes within which they poll cooperatively and share updates among each other.

So, there are three phases in cooperative polling. And this is cooperative polling, as well as independent polling, in the sense that nodes can independently increase or decrease the web sites. So, the first is the optimization phase, where a node applies the honeycomb-based optimization technique to the traffic data that it has collected from the servers.

So, any change to the polling level is communicated to the peers. And of course, nodes can change that on their own and nodes and then So, there is the maintenance phase. And then the aggregation phase is when nodes receive trade off data from other peers. So, that again, they can run the next round of optimization.

(Refer Slide Time: 59:23)



The image shows a presentation slide titled "System Management - II". The slide is part of a larger presentation, as indicated by the navigation icons on the left and the footer. The slide content is as follows:

- Initially the owner node at level $K = \lceil \log_3 N \rceil$ polls for the channels.
- A node in the maintenance phase might decide to reduce the polling level to $K - 1$
- A small wedge will form that will perform co-operative polling
- When there is a change in the polling level, some nodes need to be instructed to start or stop polling.
- Owners typically monitor the status of the nodes in the wedge, and aggregate maintenance messages.
- Upon a **failure** Corona removes the node from the ring, and on the addition of a node, Corona adds it to the Pastry ring. If the owner fails, then Corona deletes its subscription state.

The footer of the slide includes the NPTEL logo, the name "Smruti R. Sarangi", the title "Distributed Publish Subscribe Systems", and the slide number "29/34".

Motivation Design
Corona Design Corona Schemes
Evaluation Systems Management

System Management

- Each channel in Corona hashes its content to get its unique Pastry key.
- It is assigned an owner node in Pastry.
- For added fault tolerance, a key is assigned to F succeeding nodes.
- Owners receive subscriptions, and send updates to ~~the~~ all the subscribers.
- Corona manages co-operative polling through three mechanisms:
 - **Optimization Phase** : Applies Honeycomb based optimization to the traffic data collected from servers.
 - **Maintenance Phase** : Changes to polling levels are communicated to peers.
 - **Aggregation Phase** : Nodes receive tradeoff data from other peers.

NPTEL

Smrutil R. Sarangi Distributed Publish-Subscribe Systems 28/34

So, initially, the node at the owner node at level $K = (\log N_b)$. Polls for the channels. Any node in the maintenance phase, which you will recall is a second phase over here might decide to reduce the polling level to $k - 1$. In this case, a small wedge will form that will perform cooperative polling. Whenever there is a change in the polling level, some nodes need to be instructed to start or stop holding which a node will take care.

Owners will typically monitor the status of all the nodes in the wedge and aggregate all the maintenance messages. at the end over here in the aggregation phase. whenever there is a failure, Corona will remove the nodes from the ring and on the addition of a node, Corona will add it to the ring is the same as pastry. And if let us say the owner fails, then the subscription state of the owner is deleted, and a new owner will take over.

(Refer Slide Time: 60:34)

Motivation Design
Corona Design Corona Schemes
Evaluation Systems Management

Update Dissemination

- Corona has a dedicated **difference engine** that computes the difference between different versions of a file by polling the server.
- It only sends the deltas (differences) to other nodes in the polling wedge.
- Each new version of a file has a unique version number.
- When a delta is generated by a node, it shares the delta with all the other nodes in the wedge.
- If a node cannot reliably get a timestamp from the server, then it sends the delta to the owner. The owner assigns a timestamp and multicasts it. **DAK**

Smruti R. Sarangi Distributed Publish Subscribe Systems 30/34

Windows Journal

wide wedge \rightarrow low update latency
high server load

node

C^N

$N_1, N_2(t-1), N_3, N_4, N_5, N_6, N_7$

DAK

Windows Journal

wide wedge

Pen and Highlighter Settings

Pen Settings | Highlighter Settings

Color: Blue Thickness: 2pt Tip style: Point

Highlighter Settings

Color: Blue Thickness: 2pt Tip style: Point

node

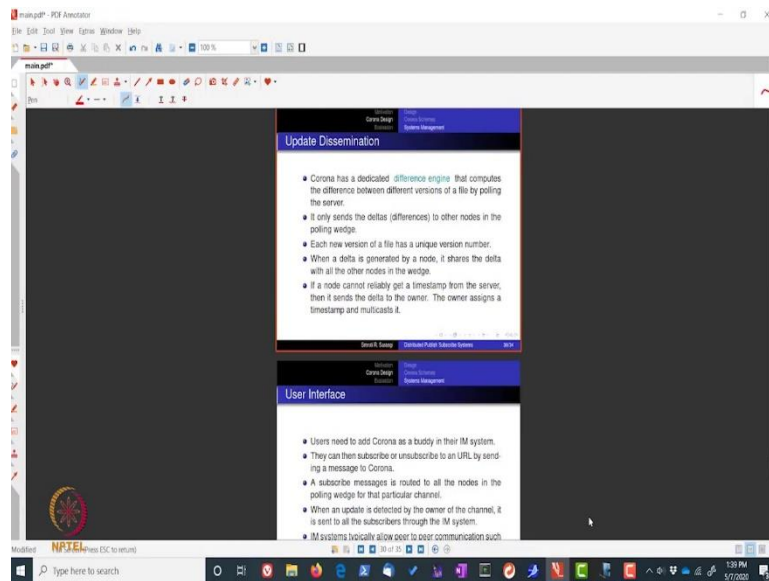
latency

load

C^N

$N_1, N_2(t-1), N_3, N_4, N_5, N_6, N_7$

DAK



So, Corona in this case, well, first, I think it is important for me to describe the way that the corona DAG will form. So, the way that it will form is something like this. That if I were to consider the pastry ring, So, initially, we will have Seville other channel here and just the owner node, it will be a very narrow wedge what will happen is this will be essentially the when the channel is being added. So, after that, of course, the servers will be polled and the node will get an idea of whether it should increase or decrease, holding level and aggregation and optimization phases.

So, let us say decides to increase it is polling them. Say we decided to increase this polling level, this is the size of the new wedge. So, this means that if the original owner was node, O, then it will point to N1, N2, N3 which are there N s wedge. Here is the interesting part of which might be hard for review for readers to readers or the pastry and corona papers to appreciate at the beginning. But this is something that needs to be explained. So, what each node actually does in pastry is that it maintains a routing table.

And in this routing table for every prefix. There are a set of nodes that match the next digit. So, some of these nodes are kind of capped in the routing table. But the important point to note is that these are not the only nodes that have 1, prefix digits matching with their binary many more nodes, but basically does not keep a list of all the nodes.

So, recall that this is the most important point hence I am repeating it once again, that let us say that between the ID of the channel. So, let us say that these are all the digits in ID of the channel. And let us say that will let us consider all the nodes. So, let me first consider the owner node that has let us say log in matching groups.

And now let us assume that it has reduced the polling level to $(\log_N - 1)$. So, when it reduces the polling level to any l , which in this case is $(\log_N - 1)$, it will essentially go to a node, it will actually go to a row in its routing table. Where it will basically see that look, show me all the nodes which have l matching prefix digits. And then depending upon the next digit, there will be a set of nodes which will be N_1, N_2, N_3 . But recall that the list is not exhaustive. There might be many more nodes, which also have l matching prefix digits, but which do not show up in the routing table.

Because the routing table typically, at least in this case does not store multiple entries. Now, assume that node N_2 on its own decides to reduce the polling level to $(l - 1)$. If it does that, it will again point to a set of new nodes which are N_4, N_5 . But given that N_1 and N_2 also match in the first l digits they will also match in the first $(l - 1)$ digits. So, one of them N_1 is showing up N_2 routing table N_2 will also point to N_1 . And so, what we will essentially and then again N_5 might decide to again changes polling level it might end up pointing to N_3 and maybe a few more nodes.

So, what we will actually have is that we will have a directed acyclic graph or a DAG and the reason will have a DAG is because the nodes independently choose to either increase or decrease the size of the polling levels. And this is what gives us a directed acyclic graph kind of structure. And the moment N_5 or N_2 and N_1 , anybody reduced the polling level, they will cut off edges and vertices from the DAG. So, of course, we will have an overall owner, and the overall owner will only know its children. And N_2 will know its children N_5 will know its children.

So, any update that is disseminated by the owner that pretty much and of course, any node will also know who its parent is in the DAG. So, let us say N_7 finds an update, its job is actually to propagate the update all the way up. There can be different variants of this, but the most common variant would be for N_7 to propagate the update all the way up to the final owner. And the final owner will then again disseminate the update to all of the child nodes So, that all the child nodes get the update.

So, this is pretty much what is being said over here. And of course, the owner dies, a new owner comes up, it will take control of the det, and maybe delete the subscription state or the new owner and requests for new subscribers. So, a lot of things are possible over here.

Corona has a dedicated Difference Engine that computes the difference between different versions of a file by polling the server, it will only send the deltas or the differences to other nodes in the polling wedge. Only if it sees it. it does not send the entire file only the deltas. And each new version of a file is given a new version number.

And definitely it will share the Delta with other nodes in the wedge. And also send it back up to the owners. This is not what i am mentioning here. This is what is mentioned of what I just mentioned, previously a slider. So, sometimes it is possible that a node is not in a position to determine if the current version of a site is new or old.

So, it cannot get a reliably get a timestamp from the server. Then what it does is it sends the data to the owner, which is supposed to be a repository of all the past versions of the site. So, the owner thinks that the timestamp is different, then it assigns a timestamp to the new version of the site and it multicast sick.

So, essentially, the new update flows through the entire DAG. So, the owner of the wedge, the original owner still has a lot of role to play because it kind of owns the DAG. And it also decides if an update is new or not. It does that and also it propagates the update to all the nodes within the DAG.

(Refer Slide Time: 67:54)

The image is a screenshot of a presentation slide. At the top, there is a navigation bar with the following items: 'Motivation', 'Design', 'Corona Design', 'Corona Schemes', 'Evaluation', and 'Systems Management'. The slide title is 'User Interface'. The main content is a list of five bullet points:

- Users need to add Corona as a buddy in their IM system.
- They can then subscribe or unsubscribe to an URL by sending a message to Corona.
- A subscribe messages is routed to all the nodes in the polling wedge for that particular channel.
- When an update is detected by the owner of the channel, it is sent to all the subscribers through the IM system. *interrupt*
- IM systems typically allow peer to peer communication such as Skype.

At the bottom of the slide, there is a footer with the text 'Smruti R. Sarangi Distributed Publish Subscribe Systems 31/84'. On the left side of the slide, there is a vertical toolbar with various icons, including a search icon, a refresh icon, and a power icon. The NPTEL logo is also visible in the bottom left corner.

How does the user interface work, So, users need to add Corona simply as a buddy as a friend in their instant messaging system. Subscribing or unsubscribing to a URL is as simple as just sending a message to Corona all the subscriber messages are routed to all the nodes in the polling wedge for that particular channel. Whenever an update is detected with the owner of

the channel, it is sent to all the subscribers through the IM system. So, here polling again is not required. This is kind of like an interrupt driven mechanism where you have consistent connections with the owner of the channel.

So, this kind of rates the server or maintaining the state Corona maintains the state like any good content provider and IM systems will typically allow such kind of a peer to peer communication with persistent network connections in something like what similar to what Skype does.

(Refer Slide Time: 68:58)

Motivation
Corona Design
Evaluation

Implementation

- Uses a standard Pastry implementation, 160-bit SHA-1 hash function
- Occasionally, it is possible that the size of a wedge might be **zero**
 - We need to then adjust the sizes of the clusters
- Corona interacts with IM systems using the instant messaging protocol - GAIM
- At the moment, the entire Corona system is trusted
- The evaluation is on a large scale deployment of Planet-Lab (large scale distributed cloud).
- Used a micronews feed collected from real life workloads.

NPTEL
Smriti R. Sarangi Distributed Publish Subscribe Systems 32/34

And So, how was this entire system implemented? Well, a standard pastry bit pastry implementation 160-bit SHA one. Of course, we have been alluding to the problem that the size of a wedge at the beginning might be 0. Then of course, we need to adjust the sizes of the clusters or the wedges by just reducing the polling level such that the wedge is not 0. Traditional instant messaging systems like GAIM are used to talk to the owner establish a connection with the owner.

So, security and trust. Well, a lot of emphasis was not given to it. We assume that all the nodes in the corona system are trusted. And the evaluation of course happened on a large cloud of machines on the planet lab cloud fair essentially micro news feeds were collected from real life workloads. And the same was simulated on a Corona based environment.

(Refer Slide Time: 70:00)

Motivation
Corona Design
Evaluation

Simulations

- System of 1024 nodes, 100,000 channels, and 5 million subscriptions
- Polling interval for 30 minutes, and maintenance interval of 1 hour
- Compare Corona-Lite, Corona-Fast, and Corona-Fair

NPTEL
Smruti R. Sarangi | Distributed Publish Subscribe Systems | 33/34

So, the system had 1024 nodes and 100,000 channels. So, this is kind of a very large system actually. So, with 100,000 channels and 1000 nodes and 5 million subscriptions, this is even much larger than many of the commercial systems that we see. But that was also the aim to see how large How much does it scale. The polling interval was set for 30 minutes and the maintenance interval for 1 hour, every 1 hour. So, there was a reconfiguration, and all the variants were compared Corona Lite, Corona Fast and Corona Fair.

(Refer Slide Time: 70:50)

Motivation
Corona Design
Evaluation

Results

Scheme	Average Update Detection Time	Average Load
Legacy-RSS	900	50.00
Corona-Lite	53	48.97
Corona-Fair	142	50.14
Corona-Fair-Sqrt	55	49.46
Corona-Fair-Log	53	49.43
Corona-Fast	32	58.75

source [1]

NPTEL
Smruti R. Sarangi | Distributed Publish Subscribe Systems | 34/34

So, for all the variants, what we actually get to see is that for the legacy RSS based system, for which we are comparing against the average update detection time, So, everything here is measured in seconds, the average updated update detection time was 900 seconds with the

average load. So, let us say the average load, let us look at that in terms of arbitrary units, that was 50. So, Corona lite, which of course, tries to minimize the update detection time subject to a constraint on the load.

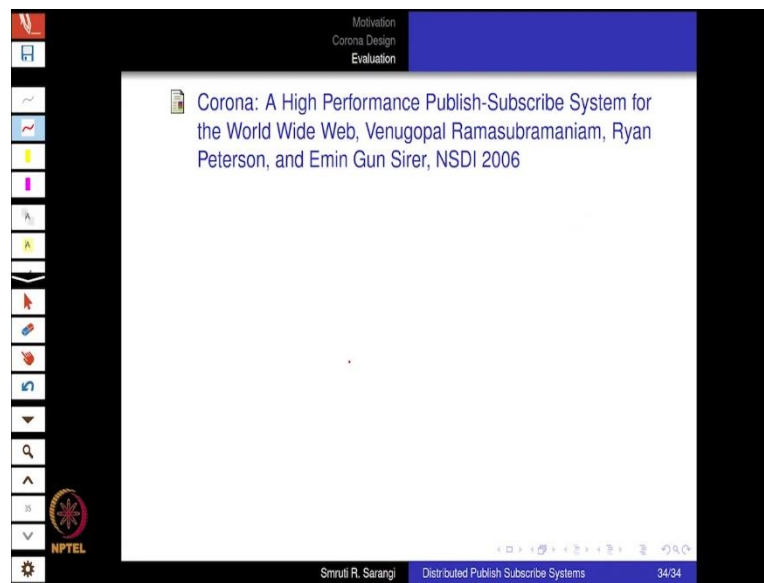
So, that immediately brought it down to 53. And the load also remained kind of similar to 50, so, the aim was never to increase the load beyond what a legacy RSS system would produce. So, it brought it down 48.97 It is 53 If I were to compare Corona fast, which you have argued that it will be faster.

So, it brought the average update detection time to 32 and the average load to 58.75. So, of course, Corona fast is the fastest and we can see it is very fast, but it increases the load at the service not much though Corona fair is something that considers the also the, the rate of change of the content itself.

So, even with a moderate load, the average update detection time was very high, the reason being that, it gives too much of emphasis to the rate of change of content to that time, but once that was kind of tempered down, and it was kind of made fair that even slow-moving content will at least get some servers. So, with this fair regime, with both the square root and the log schemes, the update detection time came roughly to the same ballpark as Corona lite, albeit with much more fairness and the load also remained very similar to the baseline RSS load.

So, what we see is that if we are looking at just straight forward update detection time we go for Corona fast. If we are looking for fairness, as well as a reduction in the server load, the best options are Corona fair square root and Corona Fair log.

(Refer Slide Time: 73:28)



So, the corona paper is around 14 years ago, it is a very classic paper, and almost all future Pub Sub systems have been built on the lines of Corona So, it is kind of provided a template for future Pub Sub systems. It was published NSDI 2006. Readers are most welcome to read the entire paper and comment about this video.