

# CORONA

A High Performance Publish – Subscribe System for the World Wide Web

Proposed by  
Venugopalan Ramasubramanian, Ryan Peterson, Emin Gun Sirer

Cornell University

Presented by  
Mohammed Shaik Hussain Ali

University of Pennsylvania

## Publish – Subscribe System?

Three Components

1. Publisher
2. Subscriber
3. Infrastructure

Two Types

1. Topic Based
2. Content Based

## Current Situation

No web protocols to automatically notify the Web content changes

"Micronews Syndication" based on naïve and repeated polling

### Resulting Problems

- Poor performance & limited scalability
- Servers face high load through active polling
- Wasted bandwidth due to sticky users
- Blockage of IP address for crossing limit of polling

# CORONA

## CORnell Online News Aggregator

Decentralized system

- High performance & scalability through Optimal Resource Allocation
- Users register through Instant Messaging Service
- Monitors the subscribed web pages
- Detects updates efficiently
- No need to change the existing Infrastructure
- Disseminates updates quickly

## Working

Computes the optimal way of implementation using a Mathematical Optimization Problem

Inputs

- Object Popularity
- Update Rate
- Content Size
- Internal system overhead (user information, dissemination, etc)

Output

- A polling schedule that achieves Global Performance Goals

Goals

- Minimise Update Latency ensuring average load on system
- Achieve target latency minimising bandwidth consumption

## Components

Channels

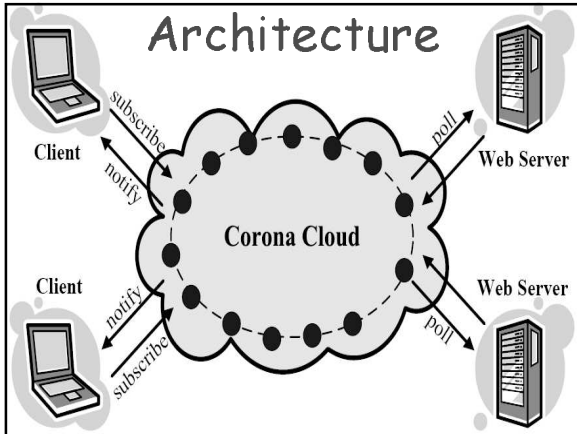
- Corona monitors a set of news feeds & web pages

Nodes

- Polling is distributed over a set of nodes (cooperative polling)

Difference Engine

- Detects whether the change is germane
- Extracts only data that has been changed
- Distributes the changes to the subscribers



Corona can be applied to any **Distributed Structured Overlay System** having uniform node degree

Corona is a **Topic Based** publish - subscribe system

Based on overlay structure as it provides

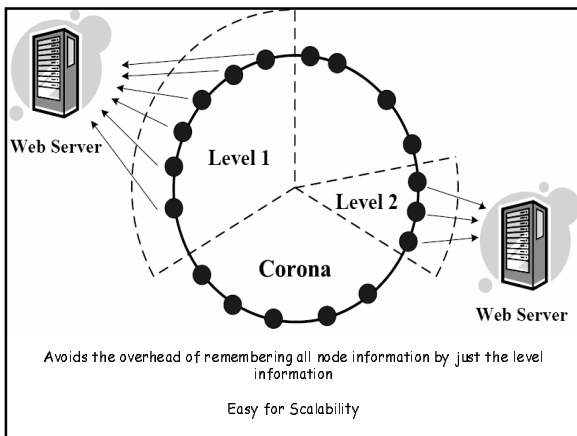
- Decentralization
- High failure resilience
- Scalability

Important key point - Uses an **Optimization Algorithm** which results in optimal resource management

Cooperative Polling

- Assigns multiple nodes to periodically poll a Channel
- Shares updates
- 'n' nodes polling with same polling interval
- 'n' times faster

$n \propto$  update performance & network load



## Analytical Modeling

### Corona Lite

Minimize average update detection time by keeping the total load constant

$$\text{Overall Update Performance} = \frac{\text{Weighted sum of update detection time of each channel}}{\text{Total number of clients subscribed to all the channels}}$$

Salient features

- Popular channel subscribers gain greater importance
- Overall global average is better
- Changes with changing load

## Analytical Modeling

### Corona Fast

Achieves target Average Update Detection Time  
By  
Minimizing the load on Content Servers

Salient features

- Opposite of Corona Lite
- Minimizes the load for a given amount of time
- Saves from flash floods & sticky users
- No increase on load even if subscribers increase
- Stays steady by changing the work load

## Analytical Modeling

### Corona Fair

Minimize Average Update Detection Time w.r.t. expected Update frequency  
By  
Bounding the load on servers

$$\text{Overall Update Performance} = \frac{\text{Update Detection Time of channels}}{\text{Update Interval of Channels}}$$

Salient features

- Both Corona Lite & Fast do not consider actual rates of change of data
- Achieves fair distribution of update performance
- The overall update performance is minimized to achieve a target load
- Two types: Sqrt & Log

## Management

- Each channel has a unique identifier based on a Hash Algorithm upon its URL
- Each channel has more than one owner nodes managing it
- Primary owner is the closest node
- Owner nodes are responsible for subscriptions, notifications & updates
- Owner nodes have information of
  - Number of Subscribers
  - Content Size
  - Update interval of site (web page)

## Management

### Periodic protocol

**Optimization Phase**  
Applies optimization algorithm on data of the tradeoff parameters

**Maintenance Phase**  
Changes in polling levels are sent through maintenance messages

**Aggregation Phase**  
Receives new values of tradeoff parameters

All phases occur concurrently

## Experiment

### One of the several experiments was

- + 60 PlanetLab Nodes
- + 150,000 Subscriptions
- + 7,500 Different Channels

### Fresh Updates

- ++ Corona Result: 45 seconds
- ++ Normal (RSS): 15 minutes

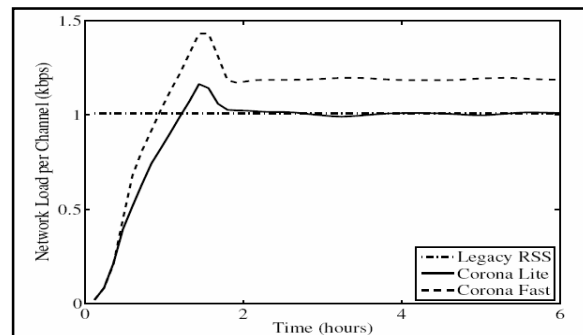


Figure 3: Network Load on Content Servers: Corona-Lite converges quickly to match the network load imposed by legacy RSS clients.

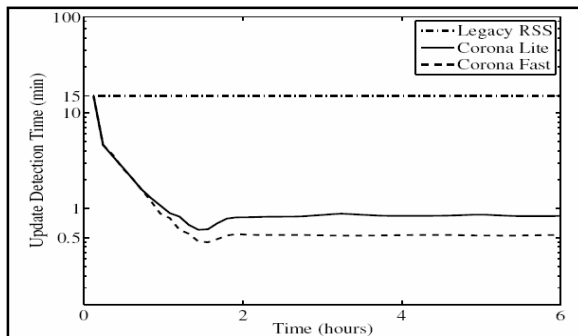


Figure 4: Average Update Detection Time: Corona-Lite provides 15-fold improvement in update detection time compared to legacy RSS clients for the same network load.

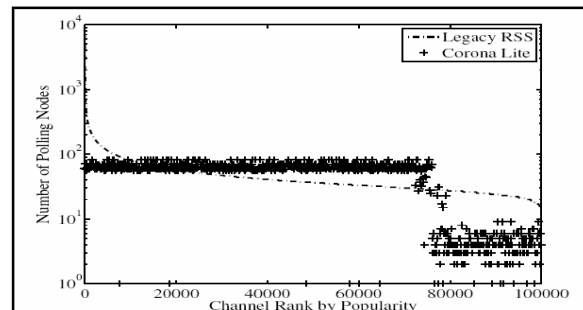


Figure 5: Number of Pollers per Channel: Corona trades off network load from popular channels to decrease update detection time of less popular channels and achieve a lower system-wide average.

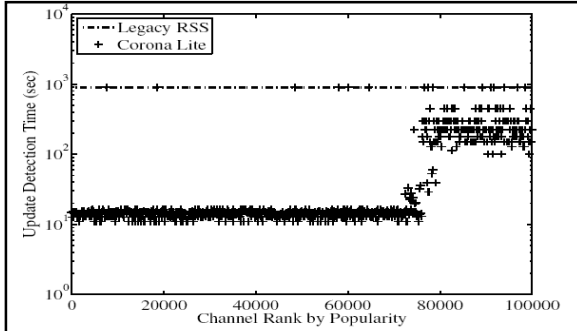


Figure 6: Update Detection Time per Channel: Popular channels gain greater decrease in update detection time than less popular channels.

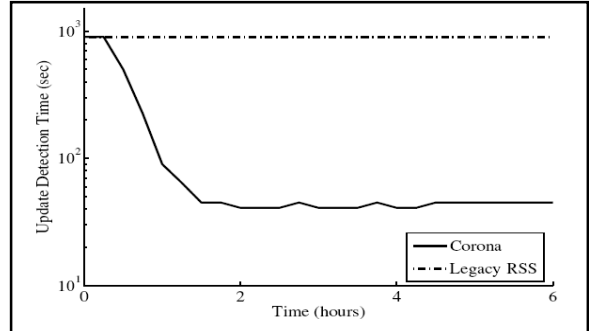


Figure 9: Average Update Detection Time: Corona provides an order of magnitude lower update detection time compared to legacy RSS.

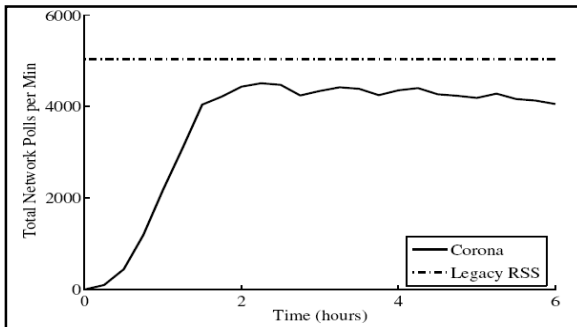


Figure 10: Total Polling Load on Servers: The total load generated by Corona is well below the load generated by clients using legacy RSS

## Summary

---

### CORONA

- is compatible with existing infrastructure
- uses mathematical optimization algorithm to arrive at an optimum resource management technique
- could be deployed over any distributed overlay structure
- achieves several orders of magnitude improvement in update latency without an increase in average load
- shields servers from the impact of flash crowds & sticky traffic