CS 237 Final Project Report

Using Peer-to-Peer networks for mission critical data communication

Satish Kotti	Siddhant Sonkar	Venkatesh SGS
skotti@uci.edu	ssonkar@uci.edu	gsoorise@uci.edu

1. Introduction

In this project, we investigate peer-based approach for improved reliability and latency for mission critical data exchange between devices (typically Internet of Things). Systems like disaster response systems and emergency alert systems need to disseminate critical information to a very large number of recipients in least possible amount of time. Since disasters and emergencies are often capable of causing immense losses to life and infrastructure, communication latency and reliability are of extreme importance. The challenges in these systems include ensuring reliability and latency in often failure prone heterogeneous networks. We use a peer-based approach for data exchange between receivers from broadcast networks. Though gossip based protocols are usually stateless, scalable with high fault tolerance, they are prone to performance degradation due to flooding of gossip messages. We discuss random walker gossip algorithms which control the flooding of gossip messages by randoming choosing a subset of peers for broadcasting messages.

The simulation and analysis of a real-time disaster response systems using gossip protocols involves the simulation of gossip network, simulation and comparison with centralized cloud-based architectures and performance analysis with random-walk gossip algorithms. In the subsequent sections, we explain our approach and design in simulating this environment and comparing the results. We further discuss the future scope of this project to simulate real-time internet topology, criticality of packet loss in failure prone networks and the effect of rapidly failing heterogenous networks on reliable information delivery.

2. Background

Previous works like [1] and [2] refer to such critical data delivery mechanisms as Flash Dissemination, a scenario involving rapid dissemination of varying amounts of information to a large audience in the shortest time. One such real-time system is *ShakeCast* [3], a service from United States Geological Survey which provides accurate and timely information about seismic events. This application collects seismic sensor data in real-time and processes it to generate a *ShakeMap* to assess the locations of impact. This information is then sent to the subscribers like city and state emergency management organization, emergency news broadcasts, hospitals, schools, etc. for immediate assessment of the impact and necessary actions, thus needing to broadcast the information to a large number of subscribers. In these kind of systems, quick delivery of messages is of critical importance as it enables the recipients to take informed and timely emergency responses.

Unpredictability, scalability and network and content heterogeneity are some crucial characteristics of these kind of situations. The exact time and location of an emergency are highly unpredictable. Therefore, it is crucial for an emergency response system to be ready to broadcast at a very short notice as very large number of recipients are spread over large areas without the need to schedule or optimize in advance. Further to this, there is also uncertainty

associated with the availability of infrastructure for the broadcast. Also, the oncoming disaster can also cause additional faults in the network topology like destroyed network cables, power outages etc. Since the audience is spread across very large areas, the heterogeneity of the network and transmission latency adds into the complexity of the situation.

An effective and easy solution for this challenge is to build a reliable and resilient network infrastructure dedicated for the purpose of emergency mitigation. However, such a solution can be highly expensive and since emergencies happen rarely, the resources are not efficiently utilized in the remaining time. Therefore, to avoid this problem, [2] uses a dissemination strategy that is more practical and feasible in the real world. The idea is to decentralize the broadcasting architecture by using a peer-based approach. In this approach, all the recipients take part in the message broadcast. This also helps with reducing the load on a centralized server and avoids being prone to a single point of failure. In addition to this, such a setting would not need excessive networking infrastructure, as this uses the existing resources at the end receivers by organizing as a large Peer-to-Peer (P2P) broadcasting network.

One of the biggest challenges of P2P architectures is being able to efficiently handle network unpredictability like failing nodes, bandwidth congestion, changing topology etc. In most situations, these systems assume normal network and host behavior like constant churn rate [4, 5]. To solve this problem of unpredictable faults, we make use of gossip-based broadcast systems [1, 2, 6, 7, 8]. However, gossip-based approaches are prone to creating an overhead of redundant messages and thus degrading the performance. For this purpose, we discuss distributed random walker algorithms [6, 7] and investigate how these algorithms can efficiently broadcast messages controlling the redundancy and performance degradation in the network.

3. Related Work

We survey multiple research works in view of the above mentioned challenges and characteristics. In this section, we explain in detail about the criticality of these challenges and they are addressed by previous works in this area.

Scalable Application Layer Multicast [10] is one solution towards the problem of one to many data dissemination. Multicast solutions separate the destination nodes from the amount of metadata that needs to be maintained. However, large parts of the internet do not support network layer multicast which make these things hard to achieve [10]. Application layer multicast solves this issue by moving to the multicast forwarding functionality exclusively at end-hosts instead of relying completely on the network. The core of the system is building a distributed tree which does not require any topological information which works well in case new additions and deletions of a node. The data delivery path is implicitly defined by the way that the tree is built. The tree is built hierarchically, that is, the set of end hosts are distributed into a hierarchy. This can also be touted as the core of the algorithm which is to create and maintain this hierarchy. Hosts closer to each other mostly end up in the same hierarchy in order to minimize stretch. However, application layer multicast sends the same packet over the same link which is less efficient than network layer multicast. Also, in the network mostly consisting of mobile devices this solution might not perform very well due to the high volume of topology changes which might result in multiple spanning tree changes which could prove to be counter-productive. Also, this

solution might tries to trade off stretch for stress in order to relieve the stress on the network. In our case, we think we can trade off stress for stretch as getting the message delivered along the most optimized path is of prime importance in case disaster response system. As the path chosen to deliver the messages is not always the shortest or fastest (depending on the metric in consideration), there is a chance of a higher delay as compared to a unicast or a peer to peer connection. Overcast[18] is another application layer multicast based solution majorly targeted towards content delivery applications.

FareCast[11] is another solution that attempts to use application layer multicast to solve the problem of flash dissemination.Similar to [10], the core of this approach is to build a spanning tree and use this tree to deliver the messages over the wide area network. They develop an ALM based tree which leans a bit more towards the peer to peer architecture. In the tree her unlike[10], a node can have multiple parents. This solution also suffers from similar problems as discussed for [10] are minimized to some extent. Due to the inherent property of the tree based structure, it presents a single point of failure but since a node can have multiple parents, the probability of this failure is reduced. But the idea itself to have multiple parents makes us lean towards a solution that follows a more peer to peer based approach.

CREW [12], ReCREW [14] approach the problem in a peer based setting. They make use of the gossip based protocol to disseminate the data. With gossip the chance of a single point of failure is greatly reduced. Gossip based protocols tend to work badly with large content but assuming that our message is not that big, we should perform well by using this protocol. This also works well with heterogeneous networks and unpredictable conditions. Also unlike the above mentioned solutions, they have an algorithm which is completely decentralized. Also, it is stateless which is an added bonus in case our devices which do not have may resources. Also, in case of an emergency situation, these networks become unstable and their behaviour is unpredictable, gossip protocols inherently due their core behaviour seem to be the ideal choice of protocol suite to solve this problem as they prefer redundancy and reliability as opposed to scalability.

In [19] the authors discuss various heuristics for flash dissemination in heterogeneous networks. The problem of flash dissemination when the nodes are allowed to have different bandwidths and latency becomes NP hard. Various solutions [22] and [23] that employ a randomized approach to solve the problem of reliable multicast are designed to work with streaming applications or handle large content. Flash dissemination requires each node to participate in the dissemination process for better efficiency. Although we lean towards these randomized approaches due to the advantages discussed earlier, these algorithms work mostly with local knowledge. As with some of the application layer multicast approaches that build an overlay tree structure have global knowledge and also maintain metadata can help in improving the performance. In this paper, the authors discuss some of the heuristics termed DIM-Time and DIM-Rank which help improve performance in a heterogeneous network. In this they make use of the knowledge (bandwidth and latency measures) obtained from a homogeneous broadcast and use this to obtain a lower dissemination time as compared to the randomized approaches.

4. Project Simulation

As shown in Figure 1, an accurate simulation of a peer-based disaster response system involves modeling of the cloud, network constraints like internet topology, packet loss, rapidly failing networks and simulation of gossip protocol between the nodes. As we are limited by the course project timeline, we cover the simulation of the gossip protocol, analysis of random-walk algorithm and its comparison with a simulated centralized server based architecture.

To illustrate the improvement in latency and reliability, we simulate this project in three parts.

1. Centralized Architecture
2. Gossip Protocol
3. Random Walk Algorithm

4.1 Centralized Architecture

In this simulation, the source node which identifies the disaster sends an alert to the cloud which takes the responsibility of broadcasting this information to all the remaining nodes. To alert the cloud of the emergency, we placed a dedicated server which listens to alerts from all the nodes. When a node sends an alert, the cloud server pushes this message to the message queue broker built using ZeroMQ [24], *Figure 2*. The broker ensures that all the subscribing nodes for the topic are informed of the emergency. The source node states the topic such as the location or type of disaster that has occurred. This mechanism therefore gives the flexibility to organize the data broadcast only to the interested parties through their topic subscription at the broker.

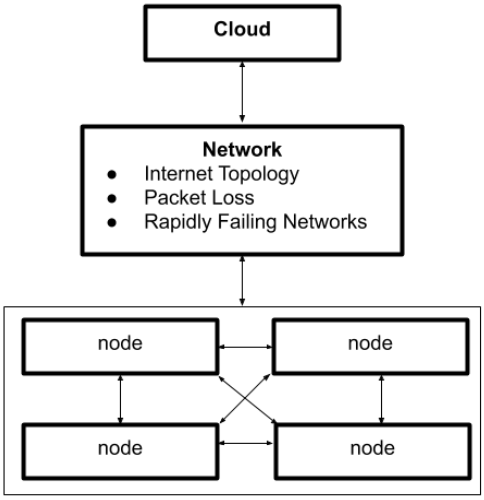

Figure 1: Gossip Architecture

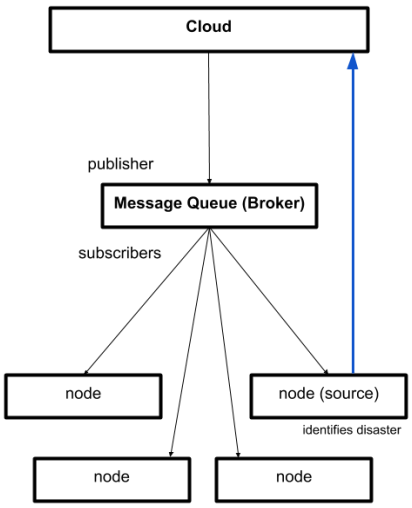

Figure 2: Centralized Architecture

4.2 Gossip Protocol

To simulate the gossip protocol between multiple nodes, we use an open-sourced package *gossip-python* [9]. The gossip mechanism between the nodes happens in two stages -

connection initialization and message spread. Every node within the network is aware of the bootstrapper node which organizes communication between multiple nodes. In our experiment, for simplicity we use only a single bootstrapper node. However, the bootstrapping node can be made use to form hierarchical clusters of nodes in extremely large networks to ensure organized hierarchical gossip communication. The following are the list of steps that are involved in exchanging data in gossip protocols.

1. Every node performs a connection handshake with the bootstrapper node to inform its network address and port information.
2. During the connection handshake, the participating nodes are also informed about each others socket address and port information to exchange data. All subsequent data communication between the nodes happen through the sockets. After successful completion of the connection handshake, the information of the new node is added to the connection pool of the node.
3. This process repeats for all the nodes that come live within the network.
4. When a new connection is formed at the bootstrapper, it informs the new node of all the network addresses available in its connection pool.
5. As the new node receives this information, it makes connections with all the new addresses received from the bootstrapper node.
6. When the connections are established, the node forwards the alert message to all the connections in its address pool. A mutex lock is acquired over the connection pool before this process to avoid redundancies and unpredictable behavior, which can be common in rapidly failing networks.
7. Upon receiving a message, every node repeats the same process informing its neighbouring peers about the message and the originating address.

4.3 Random Walk Algorithm

To integrate the random-walk algorithm within the simulated gossip network by randomly choosing a subset of peers to broadcast messages. This strategy controls the prevalent flooding in gossip networks containing large number of participating nodes. The random-walk algorithm is ideal for large networks only. In networks with limited number of nodes, the probability for reaching all the nodes within the network is quite low. This in contrast to large networks, every node has many neighbouring peers, thus the probability of each every node is sufficiently high.

5. Evaluation and Results

We simulated the results for the nodes = { 3, 5, 8, 10, 15, 20, 30 }. We used 4 laptops and 1 Android device to simulate the network to simulate a network of 5 nodes. We repeated the experiment multiple times to get a pool of latency values. We used this pool of values along with local simulation of n nodes to accurately simulate the results. For the gossip based experiments, the time taken to reach the last node is considered as the latency. Since, reliability is an important factor for emergency response systems, it is important for us to consider the latency to

reach the last node. We also used a multiplication factor of *n/10* is used to magnify the pattern of the results.

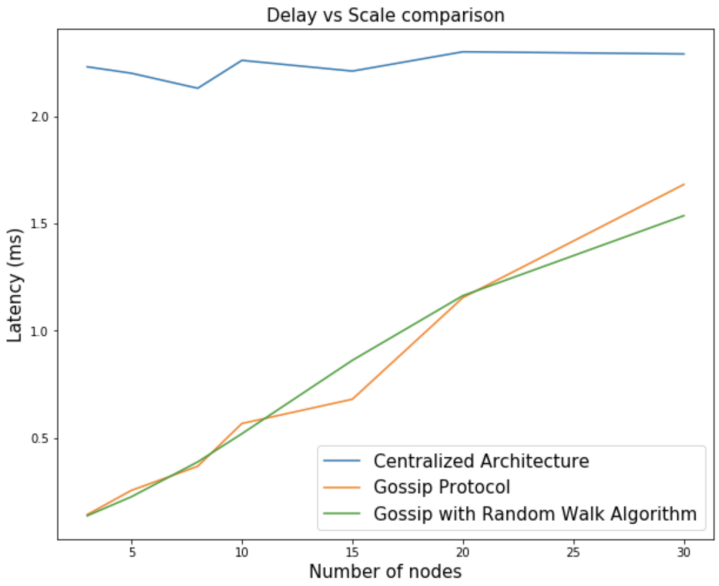

Figure 3: Comparison of results

We observed that the centralized architecture had nearly the same latency as the number of nodes increase. This is probably because the communication is one way from the server to each of the clients. Though the publish-subscribe architecture in the centralized model uses sockets, the initial alert transmission is implemented using a remote procedure call mechanism. This remote procedure writes all the messages to disk to be later picked up by the publisher. The high values for the centralized architecture can be happening due to the RPC call and wait for the messages to be written to disk. On the other side, the gossip protocol directly connects to its peers through sockets and hence results in much faster communication. Additionally, it is known that sockets have smaller headers and perform significantly better compared to REST for requests of small size [13].

In addition to the above observations, it was surprising to see that the implementation of random-walk algorithm did not have much effect on the results. This could be because 30 nodes is too small for the random walk to actually present its capabilities. We further discuss the scaling of this project for 100 nodes in the next section which will most likely illustrate the significance of random-walk algorithms conspicuously.

6. Future Work

As we implemented this project, we identified scope for possible extensions of this project, as listed below. In this section, we discuss about more directions to analyze the use of gossip-based protocols in disaster management systems.

- Varying content size and network bandwidth - emergency systems often need to disseminate heterogeneous content of different sizes. For example - images, videos, text.

Repeating the current simulation for different content sizes and in different bandwidth conditions can potentially result in interesting conclusions.

- Modeling Network Topology and constraints - the criticality of emergency systems lies in reliably delivering information in rapidly failing networking infrastructure (possibly). Therefore, it is imperative to model the network constraints like packet loss, network partitions, failing nodes and links, high network churn rate. This can be implemented using network emulation tools like ModelNet [15], Planet Lab [16] and NS-3 [17].
- The randomized approaches work with local knowledge, better heuristics can be designed to learn dynamically about the network and optimize the dissemination time by gaining some global knowledge.

It was also interesting to explore Cooja Contiki [20] and CupCarbon [21] as part of this project research and exploration. These two tools facilitate to create nodes with edge weights and to simulate a networking protocol within the nodes. However, these tools do not provide network modeling solutions. Therefore, the results from ModelNet [15], Planet Lab [16] and NS-3 [17] can be used to put into these solutions for further simulations and analysis.

7. Conclusion

From the literature study and the simulations, we were able to infer that peer-to-peer based networks perform significantly better than a centralized architecture. Though gossip networks performance degrades in large networks and with increase in the content size, the integration with random-walk algorithms can significantly control the flooding in networks. However, random-walk algorithms are of best use in only large networks, in small networks the probability of node starvation is quite high.

This project is a challenging opportunity for us to explore network middleware and to get hands-on experience with low-level socket programming. The unavailability of high quality documentation for the simulation environments for IoT middleware and network middleware is added to our existing complexities. Also it was very time intensive to understand the multi-threaded paradigm and related challenges of gossip network simulation code architecture. However, this helped us understand the challenges involved in implementing a realistic simulation of distributed systems and networking middleware.

8. References

[1] Deshpande, Mayur & Xing, Bo & Lazaridis, Iosif & Hore, Bijit & Venkatasubramanian, Nalini & Mehrotra, S. (2006). CREW: A Gossip-based Flash-Dissemination System. Proceedings - International Conference on Distributed Computing Systems. 2006. 45. 10.1109/ICDCS.2006.24.

[2] Deshpande, Mayur & Kim, Kyungbaek & Hore, Bijit & Mehrotra, S & Venkatasubramanian, Nalini. (2013). ReCREW: A reliable flash-dissemination system. Computers, IEEE Transactions on. 62. 1432-1446. 10.1109/TC.2012.68.

[3] ShakeCast - https://earthquake.usgs.gov/research/software/shakecast.php

[4] Liben-nowell, David & Balakrishnan, Hari & Karger, David. (2004). Analysis of the Evolution of Peer-to-Peer Systems. Proceedings of the Annual ACM Symposium on Principles of Distributed Computing. 10.1145/571825.571863.

[5] Padhye, J & Firoiu, V & DF, Towsley & Kurose, J. (2000). Modeling TCP throughput: A simple model and its empirical validation. Computer Communication Review. 28.

[6] Das Sarma, Atish & Nanongkai, Danupon & Pandurangan, Gopal & Tetali, Prasad. (2009). Efficient Distributed Random Walks with Applications. Proceedings of the Annual ACM Symposium on Principles of Distributed Computing. 10.1145/1835698.1835745.

[7] Grover, Aditya & Leskovec, Jure. (2016). node2vec: Scalable Feature Learning for Networks. KDD : proceedings. International Conference on Knowledge Discovery & Data Mining. 2016. 855-864. 10.1145/2939672.2939754.

[8] Mahdavi, Sedigheh & Khoshraftar, Shima & An, Aijun. (2018). dynnode2vec: Scalable Dynamic Network Embedding.

[9] gossip-python *https://pythonhosted.org/gossip-python*

[10] Banerjee, Suman, Bobby Bhattacharjee, and Christopher Kommareddy. *Scalable application layer multicast*. Vol. 32. No. 4. ACM, 2002.

[11] Kim, K., Mehrotra, S., & Venkatasubramanian, N. (2010, November). Farecast: Fast, reliable application layer multicast for flash dissemination. In *Proceedings of the ACM/IFIP/USENIX 11th International Conference on Middleware* (pp. 169-190). Springer-Verlag.

[12] Deshpande, Mayur, Bo Xing, Iosif Lazardis, Bijit Hore, Nalini Venkatasubramanian, and Sharad Mehrotra. "Crew: A gossip-based flash-dissemination system." In *26th IEEE International Conference on Distributed Computing Systems (ICDCS'06)*, pp. 45-45. IEEE, 2006.

[13] HTTP and Websockets: Understanding the capabilities of today's web communication technologies https://medium.com/platform-engineer/web-api-design-35df8167460

[14] Deshpande, Mayur, et al. "ReCREW: A reliable flash-dissemination system." *IEEE Transactions on Computers* 62.7 (2012): 1432-1446.

[15] ModelNet *http://www.sysnet.ucsd.edu/modelnet/*

[16] Planet Lab: An open platform for developing, deploying and accessing planetary-scale services *https://www.planet-lab.org/*

[17] NS-3 *https://www.nsnam.org/*

[18] Jannotti, John, et al. "Overcast: reliable multicasting with on overlay network." *Proceedings of the 4th Conference on Symposium on Operating System Design & Implementation-Volume 4*. USENIX Association, 2000.

[19] Deshpande, Mayur, Nalini Venkatasubramanian, and Sharad Mehrotra. "Heuristics for flash-dissemination in heterogenous networks." *International Conference on High-Performance Computing*. Springer, Berlin, Heidelberg, 2006.

[20] Contiki: The Open Source OS for the Internet of Things *http://contiki-os.org/*

[21] CupCarbon U-One 3.8: A Smart City & IoT Wireless Sensor Network Simulator *http://cupcarbon.com/*

[22] KOSTIC, D., RODRIGUEZ, A., ALBRECHT, J., AND VAHDAT, A. Bullet: High bandwidth data dissemination using an overlay mesh.

In Usenix Symposium on Operating Systems Principles (SOSP) (2003).
[23] Bittorrent: http://bitconjurer.org/bittorrent/.
[24] ZeroMQ *http://zeromq.org/*