ADDRESSING CHALLENGES IN BROWSER BASED P2P CONTENT SHARING FRAMEWORK USING WEBRTC

Shikhar Vashishth, Yash Sinha, Dr. K Hari Babu
Birla Institute of Technology and Science, Pilani, India
khari@pilani.bits-pilani.ac.in

AGENDA

- NodeJs, Socket.io, WebRTC => Improved efficiency and scalability
- Peer-to-peer architecture => elimination of centralized dependency, better scalability etc.
- Why not couple the benefits?
- But there are challenges!
- Let's try out implementing Chord protocol on Web Browsers and tackle the challenges

CHALLENGES

- Lack of full-fledged threading/concurrency support in the Javascript language
  - a long-running process freezes the main window
  - no provision for sleep in JavaScript
  - Unpleasant user experience
  - Web workers: communication based on event driven messaging, ensuring sequential execution is a headache

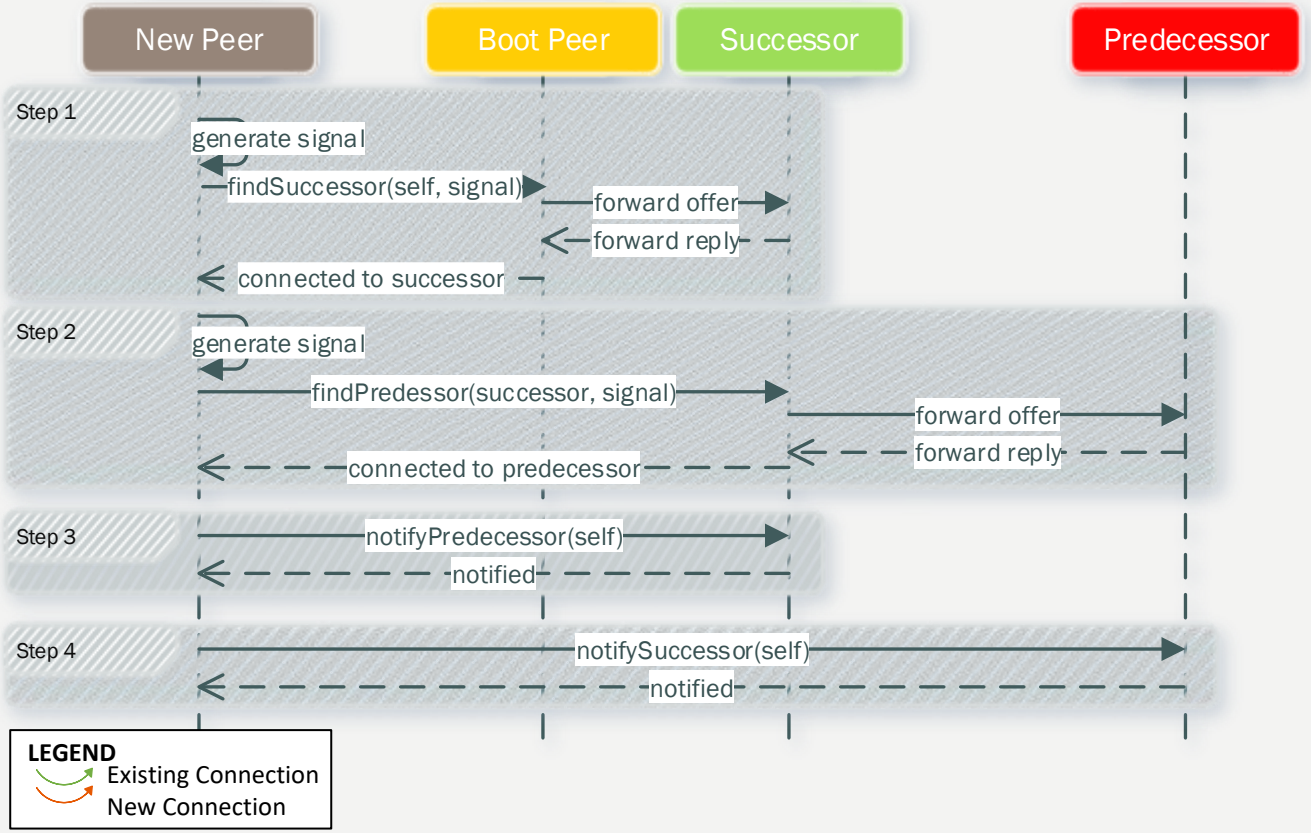
CHALLENGES

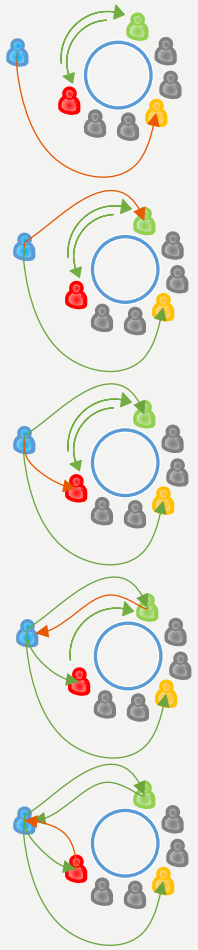
- Chord protocol's challenges
  - Dependency on bootstrap server
    - connection of new peers to the existing network
    - facilitation of handshakes and network stabilization
  - Not sufficiently asynchronous
    - reliability on synchronous loading which hinders browser interactivity because of availability of a single main thread

ADDRESSING THE CHALLENGES

- *Chord protocol procedures divided into subprocedures that can be called asynchronously*
- *Dependency on bootstrap server reduced*
  - *New peer can join network with the help of other peers in network*
  - *Network stabilization in a decentralized way*
- *Different connection strategies for different network conditions*
  - *slow STUN server vs network size*
  - preferring successor as Boot Peer vs any random peer

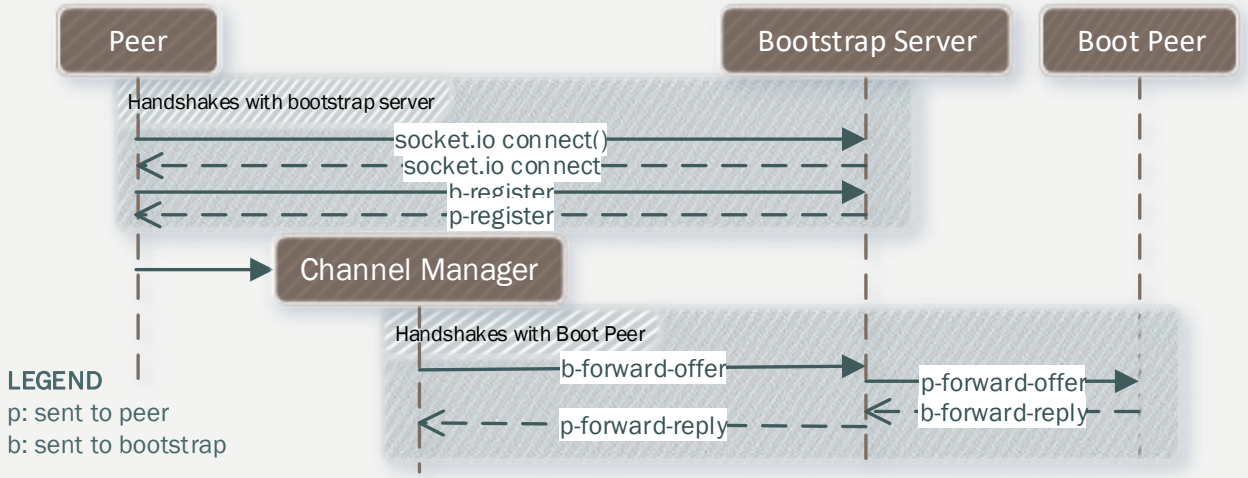
MAKING CHORD PROCEDURE ASYNCHRONOUS: JOIN NETWORK





DEPENDENCY REDUCED: JOIN BOOT PEER

- Connection handshakes
  - New peer
    - Generates connection offer
    - Sends it to the peer over the network using bootstrap server
  - Receiver
    - generates reply
    - Sends it via network
  - New peer accepts connection
- Henceforward, direct connectivity is established

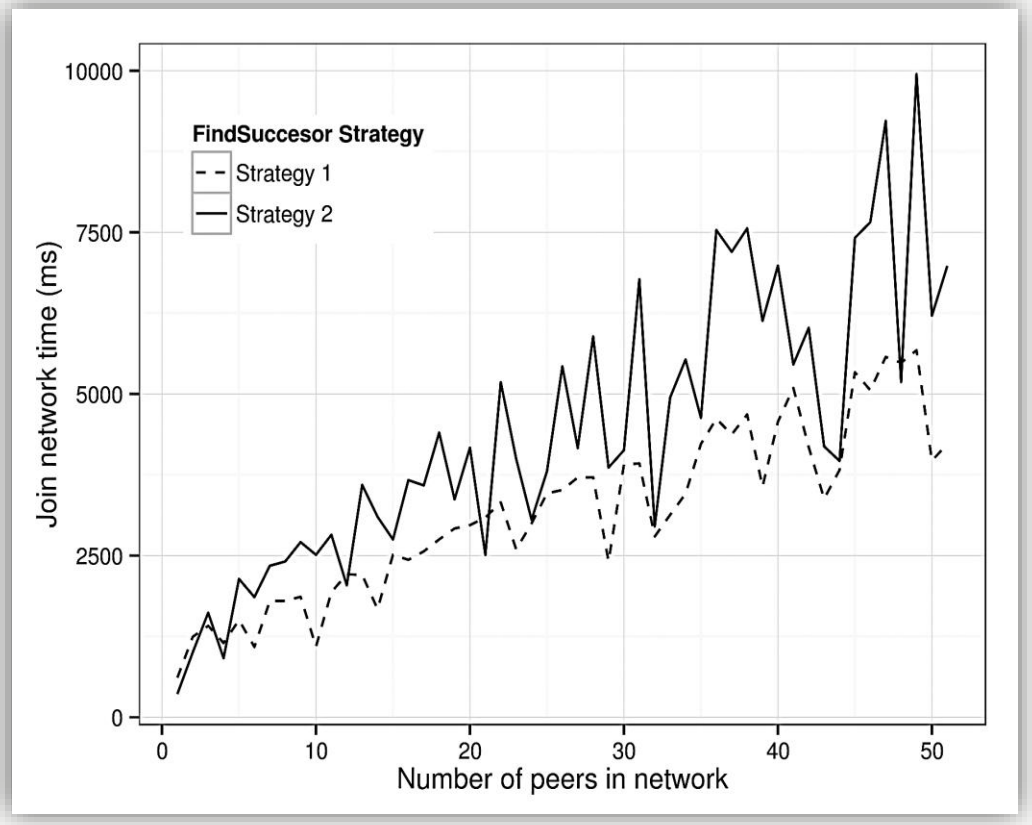


STRATEGIES TO QUERY THE NETWORK

Strategy One	Strategy Two
WebRTC connection offer along with the peer discovery query	Discovering peer in the network without WebRTC connection offer
Offer accepted or discarded based on whether connection already exists	If connection is not present, send another query with offer
Efficient when network size is large: query forward time >> offer generation time Therefore, querying the network twice becomes expensive	Efficient at times when the calling peer already has a connection with the queried peer. Therefore, no overhead of offer generation time
Efficient in scenarios where many new connections are to be made e.g., populating finger table entries after joining network	Beneficial for periodic operations for ensuring network stability. Few new connections are required to be made.
Inefficient when STUN server is overloaded because offer generation time >> query forward time	Partially better for the cases where peer already has a connection (therefore independent of STUN load)

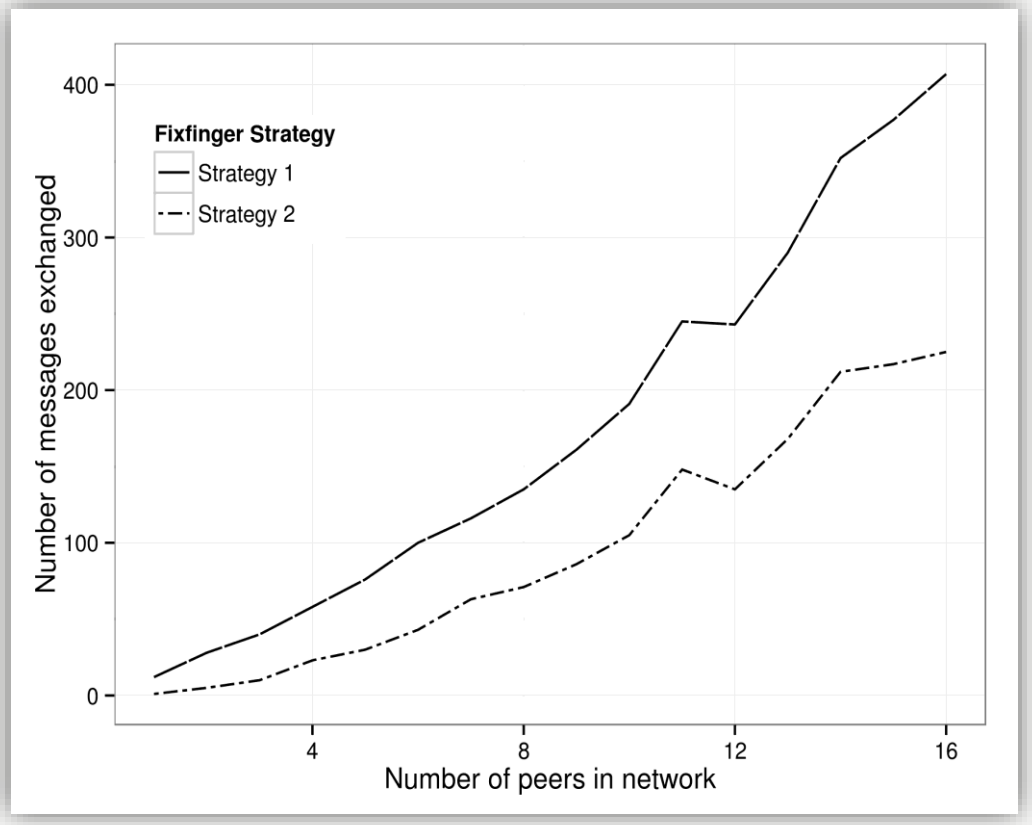
COMPARING STRATEGIES ONE VS TWO

- Small Network (0-5 peers)
  - message forwarding << offer generation time
  - ST1 ≈ ST2 (equal time taken)
- As the network grows in size
  - message forwarding > offer generation time
  - ST1 < ST2, as ST2 sends queries twice
- At times,
  - When connection already exists
  - ST1 > ST2, due to overhead of offer generation



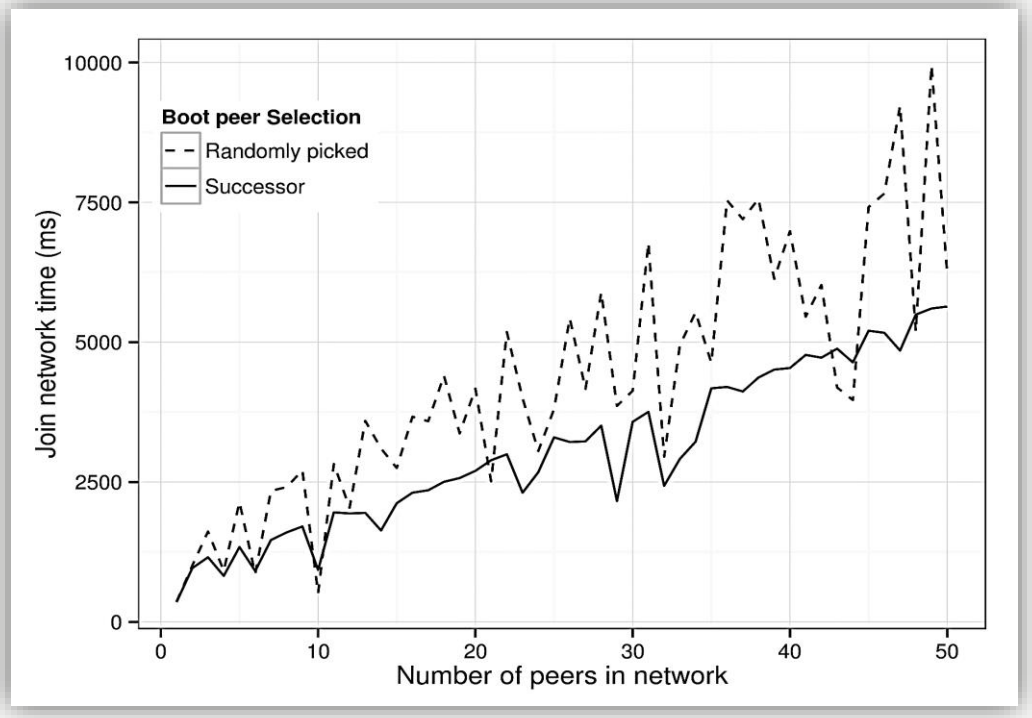
PERIODIC OPERATIONS

- Since few new connections are required to be made for periodic operations (such as for ensuring network stability),
  - ST2 performs better that ST1



PREFERRING SUCCESSOR AS BOOT PEER

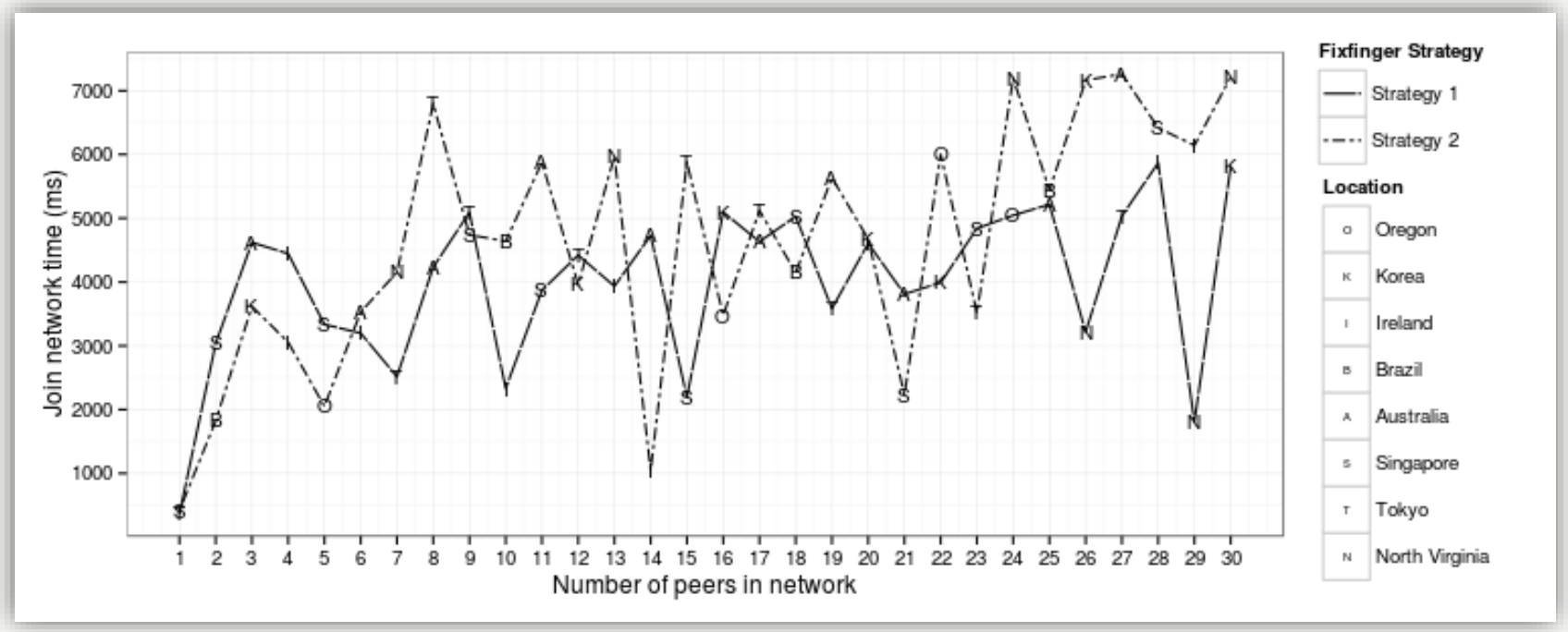
- Joining network needs connection with successor and predecessor
- Chord selects a random peer as boot peer which facilitates connection with successor and predecessor
- But in modified Chord, bootstrap choses successor as boot peer
- Therefore,
  - forming connection with random peer is not required
  - join time of new peers is reduced, thus performance of network is improved



EVALUATION ON GLOBAL SCALE

- Deployed a total of thirty peers on eight Amazon EC2 micro instances at different location across the globe
  - At Singapore, we run the bootstrap server.
  - On each instance multiple we run multiple peers.

ONE VS TWO ON GLOBAL SCALE



ANY QUESTIONS?

THANK YOU