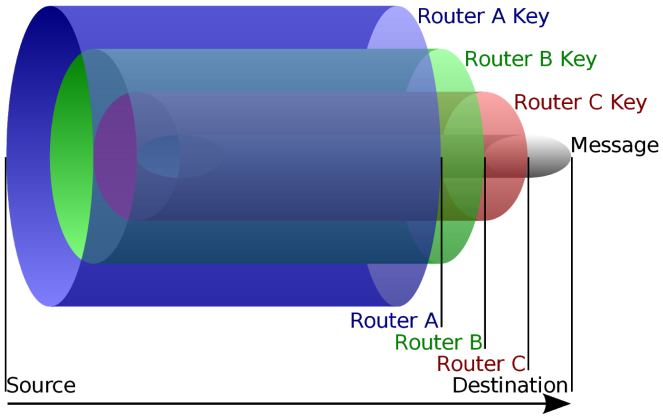
TOR File Sharing

Seongjun Lee: j2v8, Louis Mau: l5l0b, Jovan: w0c9, Andrew Chang: x7y8, Simon Plath: z4n8

Introduction:

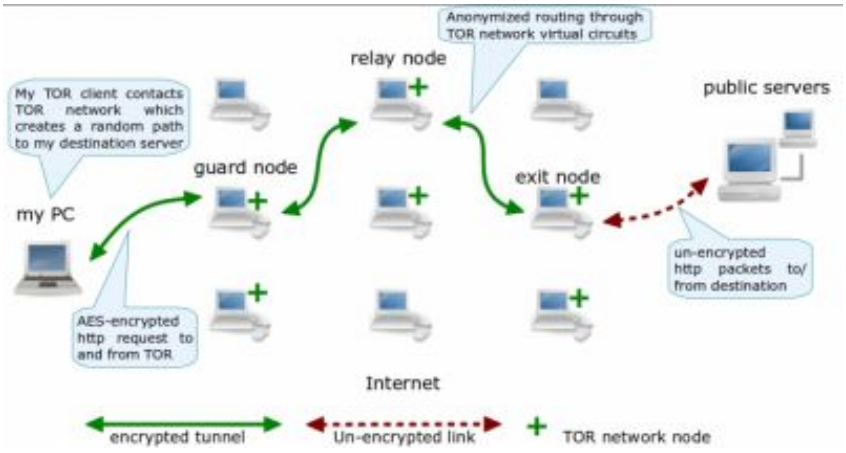
TOR is an anonymity system implemented using onion routing, providing clients a method to conceal their network identity or IP address in the network. At a basic level onion routing consists of a number of nodes (machines) forming an onion network. Data, sent in the form of a multi-layered encrypted structure, is sent from the client application along a route in the network to the destination machine or server. At each node in the network route, a layer of the encrypted data is decrypted using the corresponding private key held by that node, before it is sent forward. By this means packet content and absolute source and destination remain hidden from outside eavesdroppers, as well as nodes along the route as they only know about nodes immediately before them and after them in the route.



The project:

Our system is Peer-to-Peer file sharing using both a TOR based network, and onion routing to provide anonymity among users that are sharing or retrieving files from the network. Seeders will have the full file but the Leecher will contact multiple seeders for bits of a file. Then the leecher will be able to take all the parts that are downloaded from the anonymous nodes and create a full file. (Think of extracting a multi-part compressed file into 1 file). When leeching the leecher will need to ask the seeder nodes for a checksum or hash for the file so it knows that the chunks that are being downloaded are part of the same file. It will be on the Leecher to decide how to split the file and what file they want to grab. The goal of having this distributed system is to have file sharing that is anonymous from the outside and have a quicker way of downloading. This will have the same benefits as bittorrent where the more clients with the file

then the availability of the file increases and the more seeders you have the more leechers you can have without bottlenecking in speed.



Once the leech node asks the server to download a certain file, the server will run a flood protocol and ask all the nodes who owns a certain file and return a list of nodes to the leech node. The leech node will then use the keys from the nodes and encrypt a path for the message to take (onion routing) then send the message out to the first node. The node will then strip the first layer and see if the message is for them and then proceed accordingly.

System Details:

System Components:

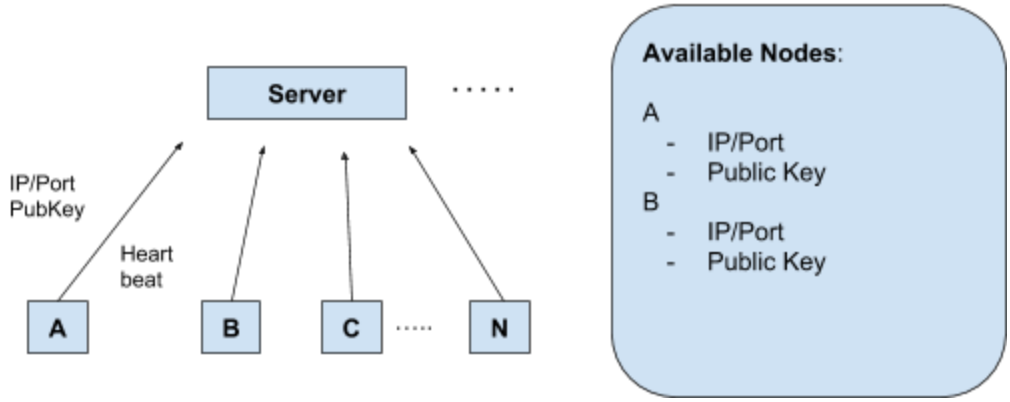
Our system will consist of a centralized server, a number of clients (nodes) which participate in the TOR network and an application controls the node.

1)Server
The server stores a list of available nodes and their corresponding IP address and public key. The server also replies search query

2)Clients
Clients query the server to determine who has the file they are looking for. Once they receive the info, they use an onion routing to retrieve different pieces of the file and recombine it once received. The number of intermediary nodes are determined in a configuration - higher number of intermediary nodes provides more security but slower transfer.

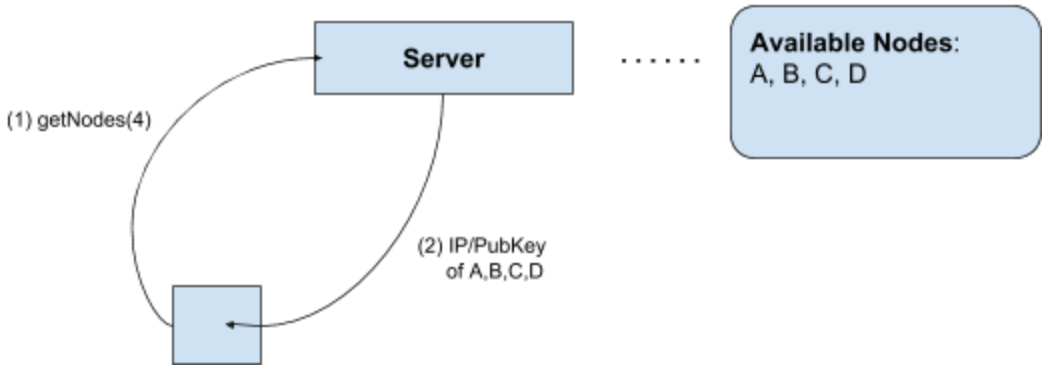
System Description:

1) Connection with Server



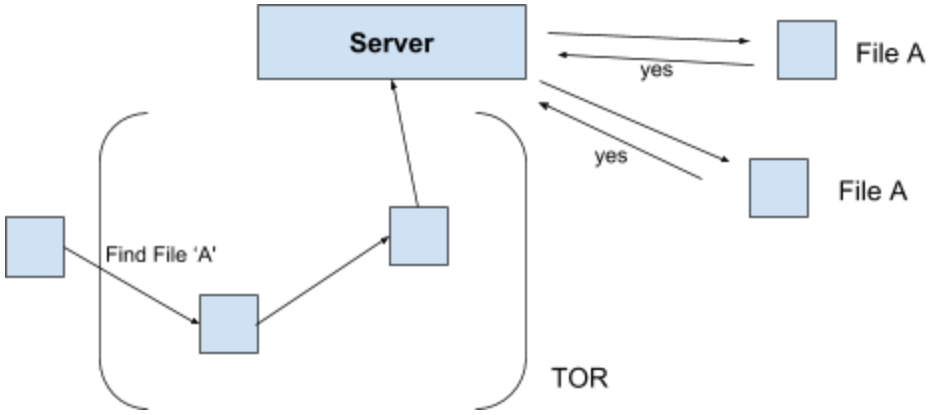
When a new node joins the TOR network it must register with the server, passing on its IP/Port and public key. The server stores this metadata and keeps track of nodes which have registered and are currently connected to it. Additionally, a heartbeat is setup from each node to the server and is used to determine disconnects on the client. The servers maintains a list of available nodes to participate in the TOR network which is subsequently updated upon disconnects.

2) Getting Nodes for Routing



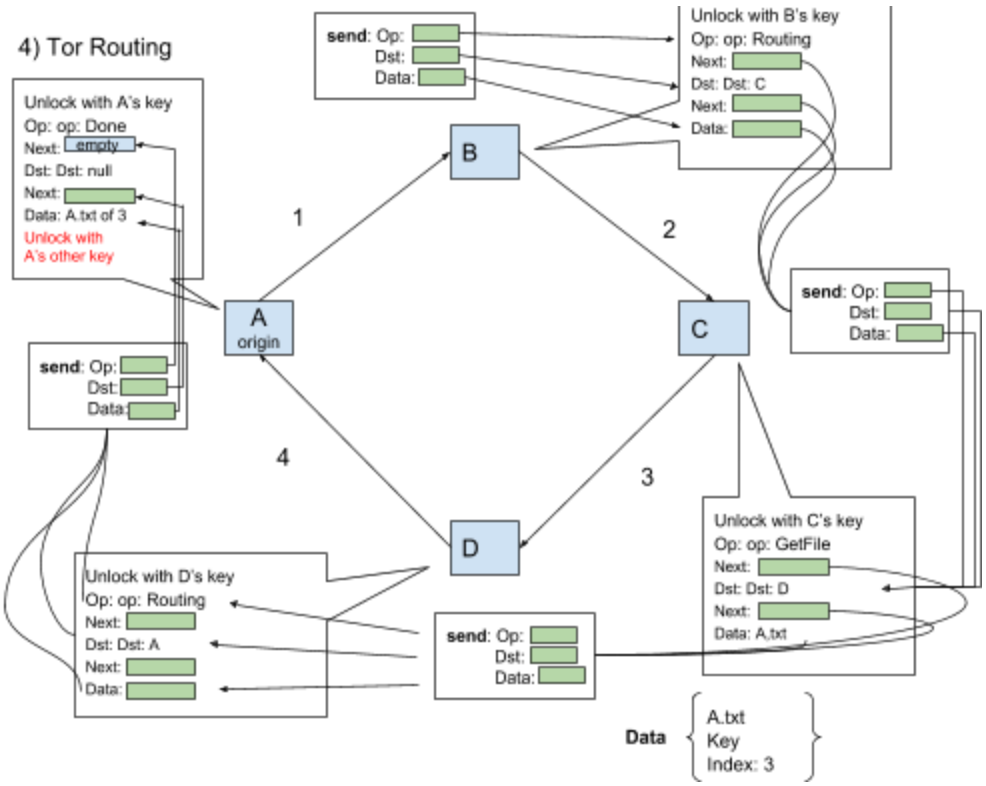
Before a client (node) requests a file chunk, they must receive a list of nodes which will form the route in the TOR network to the destination node or server. Upon requesting a list of nodes from the server, the server responds with a list of available nodes and their corresponding IP addresses and public keys. This is then used to encrypt the data and transmit it to its destination in the TOR network.

3) Search for Files (TOR)



If we search for a file, we will use the TOR network to ask the server which of the nodes have the file. The server will respond back with all the nodes that have the file. Server also provides detail of files if there are more than one file with same name but different content.

4) Tor Routing



We will use the TOR network to get the chunks of the file. In the diagram, A is the node requesting the file, C is the one with the file, while B and D are used for routing. Since we are getting small chunks of the file from each nodes, we have to make multiple getChunk requests

to many different nodes that have the file. Finally, node A needs to gather and combine all the chunks back into a single file. The layer of encryption guarantees anonymity of the transfer.

API:

getFile - starts file transfer
searchFile - ask the server to flood network for a response of who owns the file
Connect - connect to server and TOR network
Disconnect - disconnect from server and TOR network
EncryptRoute - When sending a request or a file this is used
DecryptMessage - Each node must know how to decrypt and pass on a message

Assumptions/Constraints:

- The server must run on/use Azure.
- The server must always be up, never fails
- System must scale to handle a large number of nodes
- System must handle failure in the event of nodes dying.
- System must have a minimum of 3 intermediary nodes to handle onion decryption (5 nodes including end node and server)

Challenges:

How do we know we are receiving the actual file?
The entire file is hashed and the hash is checked at the server for legitimacy.

What happens if a node that is transmitting data dies?
The received data is discarded and the node re-queries the server to get a new node to fetch from. The node discards data if it receives a number of bytes smaller than the given chunk size at the end of a transmission, or if the timeout expires.

What happens if one of the intermediary nodes dies?
A timeout occurs and the server is re-queried.

Threat Model:

Our system does not protect against any such arbitrarily strong adversary. Instead we assume the capabilities of an attacker consist of the ability to observe some amount of network traffic and generate, modify, delete and delay traffic in the network. Malicious takeover of a node in the

network is also possible by an attacker and should not compromise the anonymity of clients of the system.

The inherent shortcomings of TOR will exist in our system. If a malicious user has control of the first contact node that the client interacts with and control of the last node before the server, the malicious user will know what's being sent and who is talking to who.

The load on the server gradually increases as there are more node comes in. Centralized search query potentially cause a server failures.

Timeline:

Date	Task:
Mar 15	**Kevin - node: tor networking(peer-to-peer connection)** **Jovan - server: registration, search query** **Andrew - file System: slice/combine file feature, I/O implementation** **Louis - crypto: encrypting/decrypting message and file Chunk** **Simon - unit testing, node : heartbeat to server**
Mar 18	**Kevin,Jovan - sync with node and server** **Andrew, Louis - chunked file encryption/decryption testing** **Kevin, Louis - tor network: encrypted layered message** **Simon - application draft**
Mar 20	**Kevin,Jovan,Louis - search, getFile over network testing** **Simon, Andrew - Integration testing, application**
Mar 22,23	**Mock testing / status update / coordination -> discussion with TA**
Mar 27	**Kevin,Simon - sync with node and application** **Jovan,Andrew - end-to-end testing** **Louis - disconnection error testing**
Apr 4	**Final Report / Final testing**
Apr 9	**Demo Practice**

SWOT Analysis:

Strengths	**Weaknesses**
- All team members have worked with each other before, so are familiar with each other's work style. - We all use Slack to communicate - All team members have experience coding in golang	- Different courses and schedules - Previous failure as a result of disorganisation
Opportunities	**Threats**
- Lots of tutorials and resources on TOR and onion routing	- Course is difficult - Other coursework and exams may interfere with progress our progress on the project

References:

1. (Onion routing picture) https://en.wikipedia.org/wiki/Onion_routing
2. (TOR network picture) http://www.infosecisland.com/blogview/16290-The-Hidden-Wiki-Layers-of-The-Onion-Router-Networks.html
3. (eDonkey File-Sharing Network) https://pdfs.semanticscholar.org/242a/8ad865c8b2d40caafecd4a88a4af99b75624.pdf
4. (Peer-to-peer networking with BitTorrent) http://web.cs.ucla.edu/classes/cs217/05BitTorrent.pdf