Asian Journal of Information Technology 15 (2): 211-216, 2016
ISSN: 1682-3915

An Enhanced Fault-Tolerance Based E-Content Delivery System Using Range Queries on Peer-to-Peer Networks

[1]A. Samydurai and [2]Vasuhi
[1]Department of CSE, Valliammai Engineering College, SRM Nagar,
Kattankulathur, 603203 Chennai, India
[2]Department of Electronics Engineering, MIT Campus, Anna University, 600044 Chennai, India

Abstract: Peer-to-peer networks are gaining popularity due their decentralized nature. Among different approaches to the network, structured overlay network using tree topology is the most preferred one. However, one of the main problems of a peer-to-peer tree is guaranteeing fault tolerance in the presence of node failure. Since fault tolerance is crucial for a peer node, there is a need for a fault detection and recovery technique. In this proposal, we propose a fault detection and recovery mechanism in the tree-based peer-to-peer network. We introduce an ALIVE packet concept which is used for detecting faulty nodes and recovering from the faults when the server is overwhelmed by more number of requests from many clients. Server replication (Virtual Peer) method is adapted in the system to achieve high efficiency for range queries while providing consistency among virtual peers.

Key words: ALIVE packet, peer-to-peer, FT, overlay networks, fault recovery, fault detection

INTRODUCTION

Overlay network is a semantic layer in the overlay network is used for organizing the topology based on the content of nodes and distributed hash table is used for providing load balancing, query forwarding and lookup tables. The semantic layer is above the transport layer in the overlay network. Searching of data and convergence in a Peer-to-Peer (P2P) network is guaranteed by a virtual topology built above the transport layer in the overlay network. Peer-to-Peer overlay network (P2P) is the distributed nature of a P2P network in the application layer makes every peer communicate with other peers using the routing protocol in the layer (Wang and Li, 2003). In the case of a P2P network, each peer knows the locations of other peers and if any peer wants to communicate data with any other peer then a direct edge is formed with the other peer (Samydurai and Shanmugam, 2014a). Fault Tolerance (FT) in P2P overlay network, there has been continuous research on fault tolerance and lookup services since these two properties are crucial for P2P networks, it is necessary to provide a mechanism to discover fault nodes and recover from it (Mejias and Roy, 2007; Samydurai and Shanmugam, 2014a, b). Any fault-tolerance system must consist of two stages.

Fault detection: This is an important step in the fault-tolerance process. We first need to identify the fault nodes before recovering the network and this is the first stage of the fault-tolerance mechanism.

Fault recovery: Once fault nodes have been identified we need to provide a mechanism for recovering the network from the fault nodes. There are three ways of recovering the network from failure, namely proactive, reactive and hybrid approaches. Routing faults in P2P networks are of two types.

Fail-stop faults: The main reason for these types of faults is a link or node fault. This is one of the simplest type at fault, the peers themselves detect the fault. The only way to make the entire network stable is by isolating the faulty peers.

Byzantine faults: The non-stop nature of a fault node is the reason for this fault and these faults are also known as attacks. The attacker's main job is dismantle the entire network by corrupting the routing table and lookup services by misdirecting the messages across the peers. This fault has the ability to damage the entire network so, it needs to be properly controlled.

Need for fault-tolerance system in a peer-to-peer network:

- A P2P network offers symmetry in roles where a client may also be a server. Since, it allows access to its resources by other systems, in this situation, we require a fault-tolerant system (Lua *et al*., 2005)

Corresponding Author: A. Samydurai, Department of CSE, Valliammai Engineering College, SRM Nagar, Kattankulathur, 603203 Chennai, India

- In order to maintain the robustness of the P2P network we need an efficient fault-tolerance system (Lua *et al*., 2005)
- In a tree where every pair of nodes is connected by a single edge, a fault-tolerant system is required when any link connecting two nodes is unreliable (Jagadish *et al*., 2005)
- Real time, quality of service and fault tolerance are the essential features of a P2P system (Martins *et al*., 2008)
- Fault-tolerant systems are used for creating decentralized and P2P system (Castro *et al*., 2002)

Proposed solution: In a previous study, we used a tree-based caching technique for file sharing which stores redundant data in intermediate caches so as to decrease the time taken to access the requested data by the receiver node. But this approach does not tolerate faults in any of the nodes during data transmission. So, as an extension of this, we propose a fault-tolerance mechanism for the P2P network. Our fault-tolerance mechanism consists of fault detection and fault recovery.

Literature review: Improved the P2P ring for building fault-tolerant grids. P2P networks are gaining popularity in order to build grid systems. Among different approaches, structured overlay networks using ring topology are the most preferred ones. However, one of the main problems of P2P rings is guaranteeing lookup consistency in the presence of multiple joins, leaves and node failures. Since lookup consistency and fault-tolerance are crucial properties for building grids or any application, these issues cannot be avoided. They introduced novel relaxed ring architecture for fault-tolerant and cost-efficient ring maintenance (Mejias and Roy, 2007). Proposed (Diane *et al*., 2010) a hierarchical Distributed Hash Table (DHT) for fault-tolerant management in Peer-to-Peer Session Initiation Protocol (P2P-SIP) Networks. This paper focuses on fault tolerance of super nodes in P2P-SIP systems. The environments such as P2P-SIP networks are characterized by high volatility (i.e., a high frequency of failures of super-nodes). The majority of the fault tolerant proposed are only for physical defects. They do not take into account timing faults, which are very important for multimedia applications such as telephony. They propose HP2P-SIP which is a timing and physical fault tolerant approach based on a hierarchical approach for P2P-SIP systems.Proposed Byzantine fault tolerance of inverse de Bruijn overlay networks for secure P2P routing. Byzantine faults in a P2P system result from adversarial and inconsistent peer behaviours. Malicious peers can disrupt the routing functions in peer joining and lookup services. Byzantine attacker might scheme with each other to paralyse the entire P2P network operations. They discovered a new class of DHT-based overlay networks, called the Inverse De Bruijn (IDB) graph to enable P2P multi-path routing. The IDB provide multiple entry points and multiple routes between node pairs. The IDB overlays establish a Byzantine Fault Tolerance (BFT) framework by which secure P2P routing is guaranteed by using multiple routes to bypass faulty routes. New BFT routing algorithms are developed for secure joining and lookup operations. Consequently, fault tolerance is achieved by majority vote on data returned from multiple routes. The paper presents graph-theoretic properties of sparse IDB overlay networks and the protocols needed to establish the BFT. Proposed a P2P middleware platform for fault-tolerant, QoS and real-time computing (Lua *et al*., 2005). In this study, they proposed the architecture of Real-Time Peer-to-Peer Middleware (RTP_M), a middleware framework aimed at supporting the development and management of information systems for high-speed public transportation systems. The framework is based on a P2P overlay infrastructure with the main focus being on providing a scalable, resilient, reconfigurable and highly available platform for real-time and QoS computing. The Forward Feedback Protocol (FFP) which only routes a single copy of the message and detects the message loss and excessive delays while routing (Galuba *et al*., 2009). The routing paths are signal is failured. The binary signals with each overlay node is independently learns to route to avoid failures. The node interactions lead to the emergence of fast reliable overlay routes. This is a continuous process in which the system constantly self-organizes in response to changing delay and loss conditions. A relaxed-ring for self-organizing and fault-tolerant P2P networks. There is no doubt about the increase in popularity of decentralized systems over the classical client-server architecture in distributed applications. These systems are developed mainly as P2P networks where it is possible to observe many strategies to organize the peers. The most popular one for structured networks is the ring topology. Although, many advantages accessible by this topology, the maintenance of the ring is very expensive as it is difficult to guarantee lookup consistency and fault tolerance all the time. By means of increasing self management in the system, we are able to deal with these issues. We model ring maintenance as a self-organizing and self-healing system using feedback loops. Introduce a novel relaxed-ring topology that is able to provide fault-tolerance with realistic assumptions concerning failure detection.

MATERIALS AND METHODS

FT using ALIVE packet in tree based P2P overlay network: In the previous study, we proposed a partial caching technique for P2P file system. Now in this study, we provide fault tolerance using the ALIVE packet for P2P networks. In order to detect the faults, we make use of an ALIVE packet to indicate that a node and its associated links are reliable. When a requested peer wants to fetch data from another peer, it sends the ALIVE packet to all the nodes along the estimated path towards the destination. On receiving the ALIVE packet, the intermediate nodes will fill in the corresponding fields of the ALIVE packet and send it back to the source node. The source examines the parameters of the nodes to detect whether there is any possibility of fault occurrence. Based on this observation, it will mark that node as reliable or not. If a nodes is reliable then, it begins to the fetch the data from the destination. Once the source detects that some nodes are not reliable, it will immediately send a notification to its immediate parent node.

Architecture of peer-to-peer overlay network: The overlay network is shown in Fig. 1, consider a tree T with 'r' as the root, where N is the total number of nodes. Each node acts as a peer node. A peer is an autonomous entity with a capacity of storage and data processing. In a computer network, a peer may act as a client or as a server. A P2P network is a set of autonomous and self-organized Peers (P) connected together through a computer network. The purpose of a P2P network is the sharing of resources (files, databases) distributed among peers by avoiding the appearance of a peer as a central server in this network. This tree is a structured P2P network where each node in the first level is connected with á links and each subsequent link is connected with $á^2$ links. In this network, if any node is detected as a failure node then, its link will be replaced by its subsequent links.

Fault detection: In order to detect the faults, we make use of an ALIVE packet to indicate that a node and its associated links are reliable. When a requested peer wants to fetch data from another peer, it sends the ALIVE packet to all the nodes along the estimated path towards the destination. The format of the ALIVE packet is as in Fig. 2. The ALIVE packet has four fields as shown in Fig. 2:

- A; this field is set to 1 to indicate that the transmitter node is alive

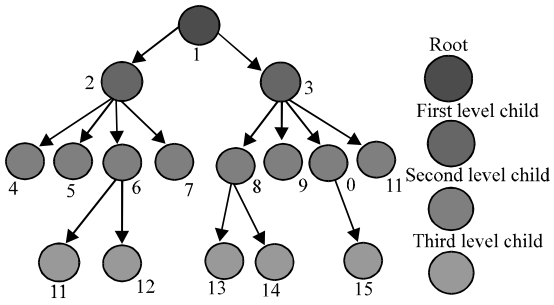

Fig. 1: Overlay network

Fig. 2: ALIVE packet

- Bit Error Rate (BER); this is used to indicate the errors in the received signal
- Buffer Overflow (BO); this is used to detect the loss of packets that occurs in the node
- Inter-Test Interval (ITI); this is the time difference between the arrival times of two sequential data

Algorithm for fault detection:

```
If (a source node S wants to send data to the destination node D)
 Then
     Retrieve a path from node S to node D from the routing table.
     Send the ALIVE packet to all the nodes in the path.
     For (all the nodes i in the path from S + 1 to D)
                    If ((ACK is not received) or (process (ALIVE) =
Fault node))
                    Then
                            Node i is faulty
                            Recover ();

                    Else
                            Continue
                    End If

     End for
 End If
```

Initially, if node 1 plans to send data to nodes 14 and 12 then it will retrieve the path stored in the routing table and before transmitting the data, node 1 will send the ALIVE packet to all the nodes in the path that is the intermediate nodes between node 1 and two destination nodes 14 and 12.

As shown in Fig. 3, the ALIVE packets are sent across both paths but the path between 1 and 12 is successful because the reply to the ALIVE packet is received by node 1 from all other nodes but node 8 has not replied to the ALIVE packet even after the threshold time, so node 8 is faulty but node 3 and from Fig. 3, it is

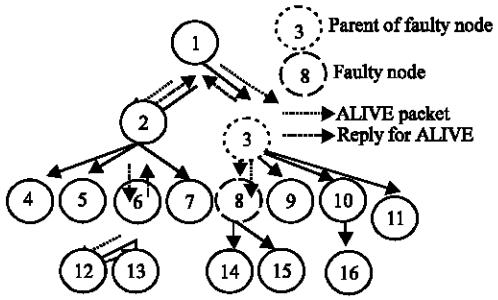

Fig. 3: Fault detection

clear that, node 3 is the one expected to be a fault node in future is indicated as it is nearing to be fault. The nearing fault of node 3 is determined after processing the ALIVE packet reply from node 3.

Recovery: The fault recovery is very crucial for getting an uninterrupted service. Whenever, a node fails and is identified as faulty in a tree-type network, the alternate parent-to-be nodes are sought to take the place of the failure node. A link is created between the node which accepts the request to become the parent and the old node's link is removed. A link with the grandfather node is also created to identify the other children of the grandfather and to create a link with them.

Algorithm for fault recovery:

```
If (a node i is under fault)
Then
    If (parent-to-be of child (i) is not empty)
    Then
                Send the request to the node in the parent-to-be list.
                If (no reply is received from the new parent node)
                Then
                        Repeat 14;
                Else
                        Form a link with the new parent node.
                        Remove the link of the old parent node.
                        Eliminate the messages sent or received
from the old parent node.
                End If
        Else
Establish a link with the grandfather node.
Become the parent of the other children of the node i.
End If
End If
```

Figure 4 shows an example of fault recovery. Since, node 8 was detected as faulty and node 3 was expected to be faulty, the fault recovery algorithm makes node 9 the parent node for nodes 10, 11, 14 and 15. According to the parent-to-be structure, the next parent of nodes 14, 15, 10 and 11 is node 9 so, all these nodes send requests which are then accepted by the parent node 9.

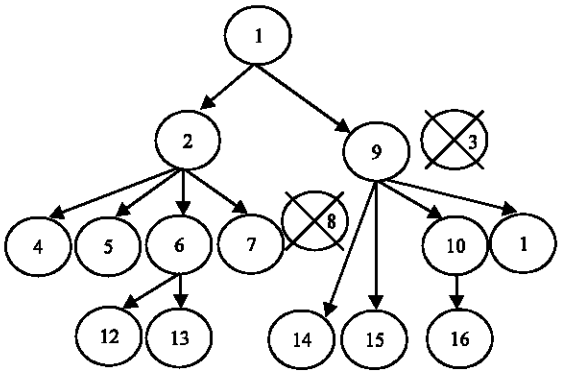

Fig. 4: Fault recovery

Algorithm for fault recovery:

```
If (a node i is under fault)
Then
    If (parent-to-be of child (i) is not empty)
    Then
                Send the request to the node in the parent-to-be list.
                If (no reply is received from the new parent node)
                Then
                        Repeat 14;
                Else
                        Form a link with the new parent node.
                        Remove the link of the old parent node.
                        Eliminate the messages sent or received
from the old parent node.
                End If
        Else
Establish a link with the grandfather node.
Become the parent of the other children of the node i.
End If
End If
```

Figure 4 shows an example of fault recovery. Since node 8 was detected as faulty and node 3 was expected to be faulty, the fault recovery algorithm makes node 9 the parent node for nodes 10, 11, 14 and 15. According to the parent-to-be structure, the next parent of nodes 14, 15, 10 and 11 is node 9, so all these nodes send requests which are then accepted by the parent node 9.

RESULTS AND DISCUSSION

This study deals with the evaluation of the experimental performance of the proposed algorithms through simulations. An NS2 simulator is used for testing the protocol. NS2 which is a general-purpose simulation tool, provides discrete event simulation of user-defined networks. The BitTorrent packet-level simulator is used for P2P networks. A network topology for the packet-level simulator is used. It is assumed that the bottleneck of the network is not at the routers but at access links of the users. Based on this assumption, a simplified topology is

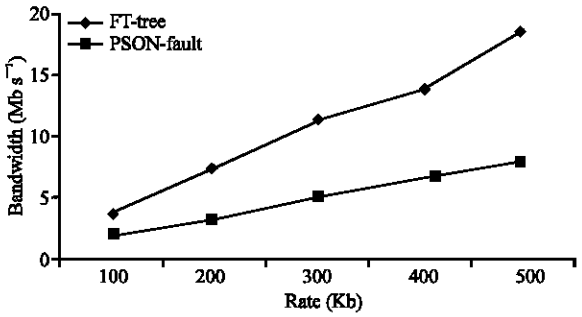

Fig. 5: Rate vs. received bandwidth

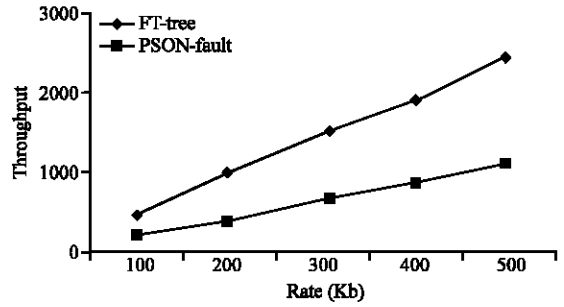

Fig. 6: Rate vs. throughput

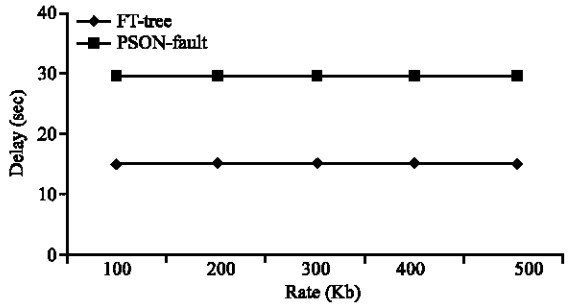

Fig. 7: Rate vs. delay

used in the simulations. The network is modelled with the help of access and overlay links. An asymmetric link connects each peer to its access router. An overlay link connects all access routers directly to each other. This helps to simulate different upload and download capacities and different end-to-end delays between different peers.

Performance measure based on rate: In the first experiment, the performance of transmission rate as 100-500 Kb. Figure 5, the received bandwidth of the proposed FT-tree method is better than the existing PSON-fault method. The received bandwidth of the proposed FT-tree method has increased by 6.03% when compared to PSON fault method from Fig. 6, it is seen that

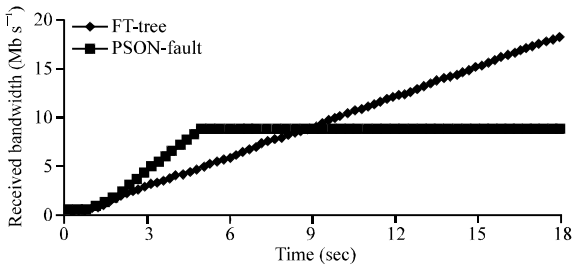

Fig. 8: Time vs. received bandwidth

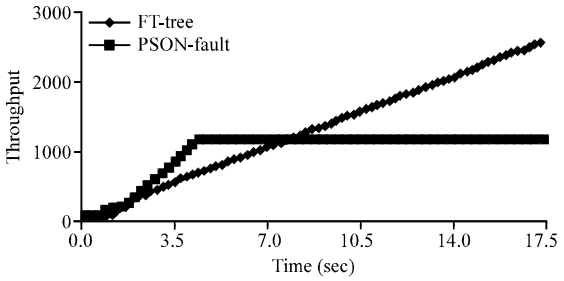

Fig. 9: Time vs. throughput

Table 1: Comparison between FT-tree and PSON-fault

Rate	FT-tree	PSON-fault
100	14.5192	28.9868
200	14.5212	28.9879
300	14.5252	28.9886
400	14.5274	28.9895
500	14.5296	28.9924

the throughput of the proposed FT-tree method is higher than the existing PSON-fault method. The FT-tree method shows the overall increase in throughput by 8.39% when compared to PSON fault method. From Fig. 7, the delay of the proposed FT-tree method is less than the existing PSON-fault method. The delay of the proposed FT-tree method is half of the overall delay shown by the PSON fault method. Thus, the FT-tree method reduces the delay by 50%.

From the Table 1 delay with respect to node for FT-tree and PSON-fault was observed. It clearly shows that FT-tree gives delay two times lesser than PSON fault.

Performance measure based on time: In the second experiment, the performance is analyzed by using the time. From Fig. 8, the received bandwidth of proposed FT-tree method is better than the existing PSON-fault method. The PSON fault method showed constant bandwidth though there were variations between 0-5 sec. The FT-tree method also showed different variations starting from 1.5 sec which was continuously increasing. Thus, overall bandwidth of FT-tree is higher than PSON fault method. From Fig. 9, it is seen that the throughput of proposed FT-tree method is higher than the

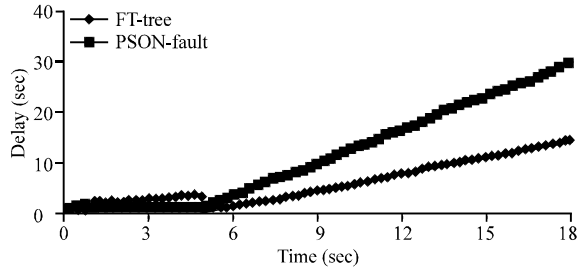

Fig. 10: Time vs. delay

existing PSON-fault method. The FT-tree method showed a continuous increase in the throughput for every time period from 1.5 sec, where as the PSON fault protocol had a constant throughput after 5.5 sec. although, initially FT-tree method provides lesser throughput. From Fig. 10, it is observed that the delay of the proposed FT-tree method is less than the existing PSON-fault method. The overall delay in FT-tree method is observed to be 167.16 sec. The PSON fault method showed a delay of 278.65 sec which is very high. Hence, the FT-tree method reduces the overall delay in the system and is better when compared to PSON fault method.

CONCLUSION

In this study, an efficient tree-based partial caching technique for P2P file sharing systems is proposed. Concerning this technique, 'k' objects are placed using the proactive segment placement approach during the deployment of the network. The leftover objects are reactively regulated by the partial caching technique. When the client sends the object request, the router calculates the hit ratio and access time of the object. Based on this calculation, the router estimates the popularity of the object. The router searches for the object first in the proactive cache proxy and then in the partial cache proxy. When the router discovers more replicas of the same object, it selects the object that has a higher popularity value. By simulation, it is shown that the proposed technique reduces the delay and improves the system performance considerably.

A mechanism for fault tolerance is also proposed. The fault-tolerance algorithm involves two sub-phases: fault detection and fault recovery. An ALIVE packet is used to detect whether a node is faulty or expected to be faulty. After the detection, a fault recovery algorithm is employed which uses a proactive and reactive approach to recover the P2P network from faults. If a fault occurs in a node, then its child node will check its next parent node and send the connection request to the next parent node. If there is no reply then, the child node will use a reactive approach, that is it will send a request to the grandfather node to obtain a new connection. From the simulation results, it is shown that the proposed approach reduces the packet drop and improves the throughput.

REFERENCES

Castro, M., P. Druschel, Y.C. Hu and A. Rowstron, 2002. Exploiting network proximity in peer-to-peer overlay networks Microsoft Res., 22: 23-24.

Diane, I., I. Niang and B. Gueye, 2010. A hierarchical DHT for fault tolerant management in P2P-SIP networks. Proceedings of the 2010 IEEE 16th International Conference on Parallel and Distributed Systems (ICPADS), December 8-10, 2010, IEEE, Shanghai, China, pp: 788-793.

Galuba, W., K. Aberer, Z. Despotovic and W. Kellerer, 2009. Self-organized fault-tolerant routing in peer-to-peer overlays. Proceedings of the Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems, SASO'09, September 14-18, 2009, IEEE, San Francisco, CA., pp: 30-39.

Jagadish, H.V., B.C. Ooi and Q.H. Vu, 2005. BATON: A balanced tree structure for peer-to-peer networks. Proceedings of the 31st International Conference on Very Large Data Bases, October 04-06, 2005, Italy, pp: 661-672.

Lua, E.K., J. Crowcroft, M. Pias, R. Sharma and S. Lim, 2005. A survey and comparison of peer-to-peer overlay network schemes. IEEE Commun. Surv. Tutor., 7: 72-93.

Martins, R., L. Lopes and F. Silva, 2008. A peer-to-peer middleware platform for fault-tolerant, QoS, real-time computing. Proceedings of the 2nd Workshop on Middleware-Application Interaction Affiliated with the DisCoTec Federated Conferences 2008, April 15, 2008, ACM, New York, USA., ISBN: 978-1-60558-204-7, pp: 1-6.

Mejias, B. and P.V. Roy, 2007. A relaxed-ring for self-organising and fault-tolerant peer-to-peer networks. Proceedings of the XXVI International Conference of the Chilean Society of Computer Science SCCC'07, November 8-9, 2007, IEEE, Iquique, Chile, pp: 13-22.

Samydurai, A. and A. Shanmugam, 2014a. Fault tolerant middleware framework to improve QoS in distributed systems. Res. J. Appl. Sci., 9: 154-159.

Samydurai, A. and A. Shanmugam, 2014b. Data sharing in a file structured QoS aware peer-to-peer system. Res. J. Appl. Sci. Eng. Technol., 7: 3995-4001.

Wang, C. and B. Li, 2003. Peer-to-peer overlay networks: A survey. Technical Report, Department of Computer Science, Hong Kong University of Science and Technology.