Variorum, Multi- Disciplinary e-Research Journal
Vol.-02, Issue-III, February 2012

An Overview of Fault Detections in Network Layer

Koshidgewar Bhasker G: H.O.D (Computer **S**cience), Degloor College, Degloor, Dist. Nanded

Abstract: Certain failures are easy to detect with the flooding scheme described in this paper. Others are not. This section will discuss which sorts of errors can be automatically diagnosed and reported, so that a network manager can then investigate the problem.

Keywords: Trusted node, Public Key List packets, Data ACK, PKL ACK, Data Packets, Pkt Size, Clk Tol

Introduction

1. Faulty Trusted Nodes.

Since the task of a "trusted node" is to generate a Public Key List Packets with the latest information, failure of a "trusted node" can only consist of one of the following:

1. Failing to send any Public Key List packet.

2. Having its Public Key List packets with highest sequence number contain incorrect information.
a. It can contain too many nodes(more than N)
b. It can leave out nodes.
c. It can contain correct nodes.
d. It can contain correct nodes with incorrect public keys.

Symptoms of such failures do not necessarily prove failure of the "trusted node", since the same symptoms can also from other causes. However as long as these symptoms can be detected by at least one non-faulty node, they can be reported and investigated.

Including Too Many Nodes in PKL

If T's latest (one with highest sequence number) Public Key List Packets contains M nodes, then any node V with parameter N set such that N < M will detect and report the problem. "the problem", in this case can be either that T is faulty, or that V's parameter N is set incorrectly.

Omitting Nodes or Reporting Incorrect Keys

If T's latest Public Key List Packets leaves out node V, or reports V with an incorrect public key, then if V receives T's Public Key List Packets, V will detect and report the problem. T's latest Public Key List Packets will reach v if a path of non-faulty nodes connects T with V, and there is no other validly signed Public Key List Packets from V with the same sequence number.

Inclusion of Nonexistent Nodes

If T's PKL contains incorrect nodes, but its report also contain all the non-faulty nodes, together with correct keys for them, this problem will not be automatically detected by the network, unless a comparison is made between T's PKL and the PKL issued by a different "trusted node". However, as long as T's PKL is correct regarding the non-faulty network nodes, the inclusion of incorrect nodes will not interfere with the correct functioning of the Network Layer.

ISSN 0976-9714

Variorum, Multi- Disciplinary e-Research Journal
Vol.-02, Issue-III, February 2012

1.1 Faulty Forwarding Nodes

Detection of faulty forwarding nodes is not easy with our scheme. The very robustness of our scheme is somewhat of a disadvantages because a problem will not be detected unless no non-faulty path exist between source and destination. For instance, if several paths exist between S and D, but failure have occurred along some of those paths, it would be desirable to detect and report the problem so that those paths can be repaired, even though they are not needed at the time. It is undesirable to detect failure only after all paths have failed, and the network is no longer operating.

Let us assume the following conditions in the network:

1. At least one non-faulty path exists between all pairs of non-faulty nodes in our network.

2. Some number of faulty forwarding nodes exists, that fall to forward packets, but do acknowledge them correctly when received from a neighbor.

This situation cannot be detected with our scheme. Every non-faulty node will successfully receive the packet with largest sequence number from every other node, and no hint of trouble will be evidenced by the state of the "Send-Flag" and "Ack-Flag" on each packet.

It is only when all non-faulty paths fall that any evidence exists that there is a problem evidence by some non-faulty not receiving the latest packet from some source. This situation cannot be detected automatically by the Network Layer with our scheme, since nodes do not know whether later packets exist, unless they receive them. However, the upper layer protocol can inform the Network Layer when it fails to get acknowledgement to its packets launched to a destination node.

In this case(when no non-faulty route exists), node by node query can determine which nodes have received the latest packet from a particular source/key pair, which will allow a network manager to have good idea of which nodes have failed. This manual query assumes that the network manager knows the topology of the network. With our scheme, the Network Layer is not aware of the identity of neighbors, so a node by node query done without knowledge of the topology will just yield a list of nodes reporting they have seen the latest sequence number, and list of nodes reporting they have not. Without knowledge of the topology, this information does not narrow down the candidate failing nodes at all – any node on either list can be either faulty or non-faulty.

20 Variant

2.1 Multiple Outstanding Packets

The above design assumes S will not generate a packet with sequence number k+1 until its packet with sequence number k has been delivered. If S fails to follow this rule, and instead issues two packet in rapid succession, its second packet may overtake its first packet, and the older packet will be dropped before delivery.

Note however, that there might be slow paths in the network, in addition to fast paths. If the source issues packet k+1 after packet k has been successfully delivered over the fastest path in the network, but while k still

Variorum, Multi- Disciplinary e-Research Journal
Vol.-02, Issue-III, February 2012

in transit over slower paths, k+1 may overtake k over some of the slower paths, causing no ill effect (in fact, causing the positive effect of limiting some redundant traffic).

Unfortunately, the source has no way of knowing when at least one copy of its packet has been successfully delivered to the destination. It can learn of that fact in approximately twice the time, since the destination can send an acknowledgement. If the source wishes to maximize its throughput, therefore, it must estimate the time of delivery. If the source issues packets too quickly, some of its packets may get lost due to being overtaken by packets with higher sequence number. But since the Network Layer is assumed to be a datagram service this is in fact legal.

A modification to our algorithm allows safer "pipelining" of packets. (Pipelining" is allowing multiple packets from the same source/key pair to be in transmit simultaneously.). The modification is to require nodes to keep, instead of a single buffer per source (and per key), m buffers per source/key pair, where m is the number of packets desired in the pipeline. If the largest sequence number seen so far by node B from source/key pair S/p is k, then node b keeps any packets with sequence numbers between km+1 and k with source/key pair S/p.

2.2 Less Persistent Data Packet Flooding.

It is not essential that a node B reacquire the database f data packets received prior to its own simple failure. Most likely, these packets have already reached their destinations, or (if the only path to the destination was through b, which was down at the time), the packets are so old that upper layers no longer need them. In fact, it is usually preferable, from the point of view of the higher layer protocols, that very old data packets do not get delivered. Public Key List packets are different, since it is essential that every node store the most recently issued PKL from each trusted node. The "persistent" design is necessary for PKLs.

In the design above, we used the same mechanism for distribution data packets and PKL packets. Since the design is overly reliable for data packets, it could be modified to be less "persistent" in the case of data packets.

The modification consists of not modifying "Send-Flag" for neighbor W on all data packets, when receiving a Restart Notification message from W. As a result, recovering node W does not receive old data packets that it had acknowledged prior to its own failure. The only case in which an old data packet should be reflooded is the case in which a source is issuing packet when memory of packets from that source with higher sequence numbers remains in the network. The mechanism of having receipt of an older looking packet trigger reflooding of the stored packet ensures that the reflooding will occur in this case.

The modification results in some bandwidth efficiency gain, since the entire database of old data packet need not be transmitted on each link to a recovering node. If the node were nonfunctional for long enough, it would never have acknowledge most of the data packets stored by its neighbors, so upon recovery the majority of the data packet database will be transmitted anyway.

Temporarily wasting bandwidth on links to a recovering node is not a very important problem, since the links were of no benefit to the network while the node was nonfunctional, and wasting the bandwidth after the node recovers is equivalent to the node having been down slightly longer. However, a more radical modification prevents wasting the bandwidth. The more radical modification consists of clearing "Send-Flag" for W on all data packets, upon receipt of a "Restart-Notification" from W.

The theory behind this proposal is that the majority of packets in the database either have al alternative path to the destination, or were generated sufficiently long ago that they will be of no use if delivered now. Only

Variorum, Multi- Disciplinary e-Research Journal
Vol.-02, Issue-III, February 2012

packets generated prior to W's recovery will fail to be flooded through W. A source cannot expect packets to reach a destination through a node which is nonfunctional. Packets generated after W's recovery will be transmitted to W

2.3 Elimination of Acknowledgments

It is interesting to note that acknowledgments are needed only as an optimization, for efficient use of bandwidth. They are not needed for correctness

The design would be correct if we eliminated all of the following:

- Packets

1. Data ACK
2. PKL ACK
3. Restart Notification
4. Restart ACK

- Flags

1. Per Packet, Per Neighbor "Send-Flag" and "ACK-Flag", for both data Packets and PKL Packets.
2. Per Neighbor, Restart Flags.

The result is a design with only tow types of packets:

Data Packets

Public Key List Packets

The volatile database consists only of the node's own sequence numbers, and the latest Data and PKL packet from each source/key pair.

The database is scanned in order. Every packet in the database is transmitted in order. If no new packet is received from a particular source/key pair, its old packet will be retransmitted every time the database is rescanned.

Since a packet are transmitted without maintaining state regarding acknowledgements, there is no need for a node to be informed when its neighbor restarts. Thus the need for Restart Notification packets is also eliminated.

This modification yields a simpler, equally robust design, but it is far less efficient in bandwidth usage.

2.4 Hierarchical Networks

An O(N) database can be impractically large in very large networks. The same trick of adding hierarchy to make a routing algorithm tractable can be used to make robust flooding practical. In this section we present a design for accomplishing flooding in a hierarchical network. We present a two level hierarchy; through the scheme can be easily extended to arbitrary numbers of levels, for even larger networks.

Topology and Addressing

The network will be partitioned into sub networks, such that each sub network is of manageable size.

Addressing will be hierarchical, consisting of two parts,

SUBNET This portion specifies which sub network the node belongs to.

NODE: This portion specifies the individual node within the sub network SUBNET.

Within a sub network, all nodes will have the same value for the SUBNET portion of their address, and all will have distinct NODE values.

There will be two types of routing.

ISSN 0976-9714

Variorum, Multi- Disciplinary e-Research Journal
Vol.-02, Issue-III, February 2012

1. **Level_1 routing –** this type of routing concern itself with all the individual nodes and links within a sub network.

2. **Level_2 routing –** this type of routing concern itself with paths to sub networks and the sub network consisting of level 2 routers, but does not concern itself with the details inside of sub networks.

Nodes that participate only in level 1 routing are known as "level 1 routers". Nodes that participate in level 2 routing are known as "level 2 routers". Level 2 routers reside in a subnet, and additionally participate in level 1 routing within the single sub network in which they reside.

The sub network consisting of level 2 nodes is known as the "level 2 sub network".

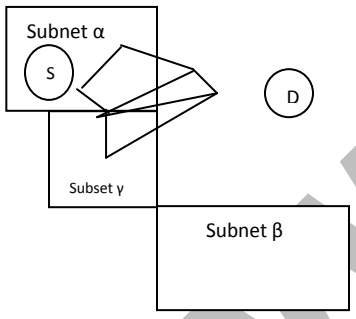
General Routing Pattern

When source and destination nodes of a packet are in the same sub network, the packets are flooded only within the sub network. When source and destination are in different sub networks,

1. First the packet is flooded throughout the source sub network.

2. Next the packet is picked up by each of the Level 2 routers in the source sub network, and signed.
3. Each of those Level 2 routers floods their copy of the packet throughout the Level 2 sub network.
4. Each of the Level 2 routers in the destination sub network pick up and sign each copy of the packet, and flood them within the destination sub network.

Thus the destination will receive multiple copies of the packet, equal to the product of the number of level 2 routers in the source sub network and the number of level 2 routers in the destination sub network.



For example in the picture above, source s generates a packet with destination d. The SUBNET portion of s's address indicates that s resides in subnet α . The SUBNET portion of d's address indicates that d resides in the subnet β.

The packet is first flooded throughout subnet α, using the resources and signed by the public key of s. Since the destination address indicates the destination is in a different subnet, all level 2 routers residing in subnet α, i.e. A,B and C, pick up the packet for flooding throughout the level 2 sub network.

Each of A,B and C independently supply their signature and sequence number to the packet. Since the node in the level 2 subnet and the nodes in subnet β have no knowledge of the ultimate source s, s's identity,

ISSN 0976-9714

Variorum, Multi- Disciplinary e-Research Journal
Vol.-02, Issue-III, February 2012

sequence number, and signature are at this point irrelevant pieces of information in the packet, from the point of view of the Network Layer.

Thus each of A, B and C independently overwrite the header fields:

- Source node
- Sequence number
- Public key
- Packet signature

With their own ID, sequence number, public key and signature.
At this point, the packet becomes three independent packets as for as the level 2 subnet can detect.

When one of the level 2 routers in subnet β (E or F) receives a flooded packet destined for subnet β (as indicated by the SUBNET portion of the "destination node" field in the packet header), it picks up the packets for flooding throughout destination subnet β. As before since node A, B and C are not known within subnet β, the fields overwritten by A, B and C (source node, sequence number, etc.) are now irrelevant inside of subnet β. Thus each of E and F independently overwrite the same header fields.

- Source node
- Sequence number
- Public key
- Packet signature

With their own ID, sequence number, public key and signature.

Since E and F cannot correlate the packets they receive from A, B and C as al having originated from the same ultimate source, (because they cannot remember more than a constant number of packet from each level 2 router, and the packet from A, B and C resulting from s's original packets may arrive at different times), each of E and F will originate three separate packets into subnet β**,** one for each packet of A, B and C

Databases: As in the non-hierarchical flooding scheme, nodes within a sub network keep stat about all the other nodes in their own sub network/ In other words, public keys are kept for, and buffers are reserved for, each other node in the sub network. Level 2 routers participate within a single sub network, but in addition keep state about all the other level 2 routers. A level 2 "trusted node service" broadcasts public keys for all level 2 routers, within the level 2 sub network. Each level 2 router keeps public keys and buffers for each other level 2 routers.

Why this works

- Flooding within a sub network (source ID and Destination Id have identical "SUBNET" fields) works identically with flooding in a nonhierarchical network.
- When Source and Destination are in different subnets, each level 2 router acting on behalf of the source sub network must guarantee fairness for all sources within that sub network. Then each source in the source sub network is guaranteed some sources within the level 2 sub network (1/N of the resources guaranteed to the level 2 router, where N is the number of level 2 routers).

ISSN 0976-9714

Variorum, Multi- Disciplinary e-Research Journal
Vol.-02, Issue-III, February 2012

Each level 2 router R which introduces level 2 traffic into the destination sub network must guarantees fairness to each level 2 router (each "source" in the level 2 sub network), for the introduction of traffic into the destination sub network. Then each level @ router will be guaranteed 1/M of the resources guaranteed to R whit the destination sub network where M is the total number of level 2 routers ID. Key pairs within the level 2 sub network.

This assures that source sub network is guaranteed access into the destination sub network. Thus the fraction of bandwidth guaranteed to source node S within a foreign destination sub network is the fraction of bandwidth guaranteed per node in the source sub network, times the fraction of bandwidth guaranteed to each level 2 router in the level 2 sub network, times the fraction of bandwidth guaranteed per node in the destination sub network.

2.5 Flooding Without Network Layer Cryptography

It is interesting that a flooding scheme can meet the Byzantine robustness criteria without Network Layer Cryptography and without O (N) buffers. We present here a scheme that the oretically accomplishes the robustness goal without using Network Layer Cryptography, but is totally impractical due to the negligible performance it achieves.

This scheme requires use of reasonable accurate elapsed time timer (as opposed to globally synchronized clocks). Some threshold say 10%, is set, such that it a node's elapsed time clock is not within that percentage of true elapsed time, the node would be considered faulty.

The robustness achieved by this scheme is " A packet from non-faulty source A to non-faulty destination D will have high probability of reaching D provided that at least one path of non-faulty processor and links connects A and D, regardless of the number of other faulty components in the network"

Priori Knowledge

The manually configured information required at each node consists of:

- H- the maximum path length in the network
- PktSize – the maximum sized data packet to be processed
- Nbrs – the maximum number of neighbors of any node in the network
- ClkTol – the maximum allowable ratio of measured times between two non-faulty processors. To ensure this, we require the ratio of time measure by any non-faulty processor to true elapsed time to be between 1/ClkTol and ClkTol.

A data packet contains the following:

Source node	The Identity of the source node
Destination node	The Identity of the destination node
Remaining hops	The number of hops further this packet is allowed to traverse
User data	To be delivered, but not interpreted by the Network Layer.

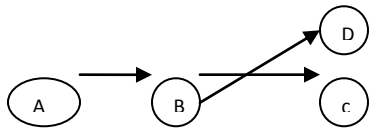
Each node keeps H buffers, each of size PktSize. Each buffer is reserved for packet with a specific hop count. A packet occupies the buffer corresponding to the hop count specified in the packet's "remaining hops" field.

Since routing is via flooding, each packet must be transmitted to all neighbors except the one from which it was received. Thus associated with each buffer is flag for each neighbor, indicting whether the packet occupying the buffer still needs to be transmitted to that neighbor. When all flags are clear, the buffer is free for acceptance of another packet with the specified "remaining hops".

ISSN 0976-9714

Variorum, Multi- Disciplinary e-Research Journal
Vol.-02, Issue-III, February 2012

The M/S scheme, with per neighbor fairness added, assures fairness and packet progress in the absence of Byzantine failure. However a single Byzantine failure in the network can cause zero throughputs on a non-faulty path in the network.



In the picture above, the Path A-B-C is non-faulty , and A has a packet with "remaining hops" of 3, waiting to be transmitted to B. Non-faulty B has a packet with "remaining hops" of 2, waiting to be transmitted to D , who is faulty and will never accept a packet from B. Thus A will never be able to transmit to B even though B is non-faulty, with this scheme. To prevent Byzantine failures from blocking the progress of a packet through a non-faulty path, we must also add timers. This scheme can be shown to meet the correctness condition by induction:

Suppose a non-faulty path $A_1,A_2……A_k$ exists between non-faulty processors S and D. Assume that the path is exactly H long, and that S initialized "remaining hops" to H. Since D, assuming it is non-faulty, will process each packet within 1 time unit, and since D has at most Nbrs neighbors, D will accept the packet within Nbrs time units(times the fudge factor of ClikTol).

Assuming A_k is non-faulty, it will forward each packet occupying the buffer for "remaining hops" of 1 each time the neighbor to which it is queued request it. Since a neighbor is required to request a packet with "remaining hops" of 1 within Nbrs time units, from each neighbor A_{k-1} can expect that A_k will be reay for receiving the packet within an extra factor of Nbrs.

If the path is actually shorter than H, the proof still holds. The only consequence of setting "remaining hops" higher than necessary (though a value greater than H is illegal and would be discarded), is that the timers are longer than would be necessary for the S to D path.

This scheme has the obviously undesirable property that the throughput on a non-faulty path of m hops is only guaranteed to be ne packet for every $(ClkTol*Nbrs)^m$ time units, making this scheme clearly without practical utility.

An additional point of note is that the scheme does not necessarily yield a storage advantage over the cryptographic, per-neighbor scheme discussed in the main portion of this chapter, since H is likely to be O(N). In order to meet the robustness requirement that non-faulty node pair A and b should be able to communicate provided that any non-faulty path between there exit, then H must be at least as large as the longest possible path between any pair of nodes in the network. Thus H is quite likely to be very close to N-1, the longest possible path in any network of n nodes

References

1. Computer Networks by Andrew S. Tanenbaun

2. Data and Computer Communications By Willian Stallings

3. Internetworking with TCP/IP, Principles, Protocols Architecuture by douglas E. Comer

4. Ruting in Computer Networks by Frant H. and W.Chou

ISSN 0976-9714

Variorum, Multi- Disciplinary e-Research Journal
Vol.-02, Issue-III, February 2012

5. New Directions in Cryptography by W.diffie and M.Hellman

6. D.Bertsekas and R.Gallager, "Data Networks", Prentiss-Jall

7. E.W.Dijkstra, " A Note on Two Problems in connection with Graphs", Number. Math. Vol-1.