System Software Support for Router Fault Tolerance

Muhammad Azam*, Nicholas Ioannides*, Mark H. Rümmeli**, Igor Schagaev*

* Faculty of Computing, London Metropolitan University, 166-220 Holloway Road, London, N7 8DB, UK
** IFW Dresden, P.O. Box 270116, D-01171 Dresden, Germany

Email: {MUA0504, n.ioannides, i.schagaev}@londonmet.ac.uk
Email: m.ruemmeli@ifw-dresden.de

Abstract: Improvements to router reliability using the generalized algorithm of fault tolerance (GAFT) are presented using time, structure and information types of redundancy. Router functionality and options to tolerate faults are discussed and analysed. Separation of toleration of malfunction and permanent faults is discussed in terms of their impact on system reliability. A scheme of reliability improvement for router is proposed using system software recovery points, with brief explanation of their implementation.

Keywords: Router, Router Faults, Fault Tolerance, Generalized Algorithm of Fault Tolerance, Reliability

1. INTRODUCTION

Communication networks are made out of nodes and links. Both items include hardware and software components and are expected to provide communication through these networks in a reliable and transparent manner. The reliability of the network depends on how well these components are able to negate or tolerate a fault, which could be a malfunction or a permanent fault. A system which provides full functionality for its application and can recover transparently from predefined faults is called a fault tolerant system (Schagaev and Zalewski, 2001).

Networks are, in general terms, capable to detect any irregular situations which may result in malfunctions or permanent faults. Depending on topology and the protocols used, when an actual fault does occur multiple alarms could be generated by a number of network elements (fault-detection phase). One of the major functions of modern network management is to determine the root cause of any fault which may occur and perform fault detection, fault localisation and fault isolation processes.

The most important traffic regulating device in a communication network is the router. The sustainability of router hardware and software will determine the reliability level of a particular network. Depending on the character and strength of the fault, the network’s downtime may last from a few seconds to several days. Very often such network failures are caused by hardware malfunctions.

The appearance and manifestation of a hardware fault may cause errors in data transmission. The normal way to recover an erroneous reception of data is to request retransmission of the particular packet(s). However, the fault might have happened inside a router during the process of path computation/determination and/or packet encapsulation. In this case, a packet could be corrupted whilst being read from the input buffer, written into the processor cache, processed, or written into output buffers. Thus the overall reliable operation of a router becomes critically important for network efficiency and performance.

2. BASIC IP ROUTER STRUCTURE AND FUNCTIONS

Routers consist of basic components like network interfaces, processing modules, buffering modules (memory), and an internal interconnection unit (switch fabric or bus). Inbound network interfaces receive packets and forward them to the processing modules where they will be processed and stored in buffering modules. They will then be forwarded to the outbound interface, determined through the process of path determination, via the internal interconnection unit in order to be transmitted to the next hop in their way towards their destination. All packets from all connected network interfaces must be processed, buffered and then directed.

Figure 1 shows the basic architecture of an IP router (a) and its routing components (b).
In the basic architecture diagram, Figure 1(a), one can see the controller card which also holds the central processing unit (CPU), the router backplane and interface cards. The typical task of a CPU in a router is to perform functions such as path computation/determination, routing table maintenance, and reachability propagation (route processing). It is independent of the type of routing protocol used and can run any routing protocol as needed.

In the routing components’ diagram, Figure 1(b), one can see the fundamental functionalities in an IPv4 router which are route processing, packet forwarding and any router special services. When searching for a route in the routing table the longest prefix that matches the destination address of the packet is needed to be found thus making routing tables quite large and time consuming when a particular route is determined. This results in routing tables acting like bottlenecks. Instead, classless inter-domain routing (CIDR) (Fuller et al. 1993) is used to summarize a block of same class addresses into a single routing table entry with the final path determined much closer to the destination. When a packet arrives, a routing table lookup is performed by the router using the packet’s IP destination address. This process returns the best-matching routing table entry, which identifies to the router the IP address of the next hop for the specific packet and from which interface to forward the packet out.

The interface cards employ adapters to control the forwarding of any inbound and outbound traffic before and after processing whereas the transferring of the packets between these cards is done by the router backplane. These cards may also temporarily store routing table entries or can have extensive packet processing capabilities. In addition, they perform protocol and time critical, quality of service (QoS) processing functions that lie in the path of data flow thus classifying packets going through a port with varying QoS levels. These simultaneous exchanges of control and data massages between network interfaces are possible due to a high performance switch which provides interconnection between them.

Very high traffic loads result in bottlenecks due to limitations in processing speed, memory size and switching bandwidth. Increasing the processing speed and memory size can decrease this problem. However, to really deal with this problem a completely distributed architecture along with a switch fabric is necessary where network interfaces are designed to process a route locally thus increasing the overall performance of the router (Koufopavlou et al. 1994).

3. ROUTER FAULT TOLERANCE AND RELIABILITY

As the Internet is growing faster than ever and used for real time and safety critical applications, the need for reliable links and interconnections between reliable routers is increasing dramatically (Thomson et al. 1997). Strength and availability are quickly becoming very important parameters in the design of the switch fabric. Router performance and QoS must be accompanied by strictly dependent router reliability. But to achieve availability and fault tolerance the crucial components of routers must deliberately use added redundancy along with an implementation algorithm (Aweya, 2009). The implementation and adaptability of a generalised algorithm of fault tolerance, known as GAF (Schagaev and Zalewski, 2001), is analysed below for the specific need.

3.1. Router faults

The possibility of faults must be considered during the design, development and manufacture phases of a router. The primary mechanism for diagnosing these faults is non-concurrent testing which places a part into a special testing mode.

Operational faults are classified based on their source:
- Environment (temperature variation, EMI and radiation);
- Operation mode (very-low-voltage operation and slip-ups of communication);
- Process technology (Benini and De Micheli, 2007) (parameter spreading, defect density and failure rates);

Physically, faults are classified as permanent, intermittent, and transient (Weaver and Austin, 2001). From the system’s viewpoint, the nature of a hardware fault is different resulting in the fault being considered either permanent (hard) or temporary (soft) called malfunction. The ratio of malfunctions to permanent faults is roughly 10^5 for modern technologies. The main reasons behind permanent faults are electro-metal migrations, hot electrons and latch-ups caused by a unity gain in the bipolar transistor structures present in the CMOS layout (Weaver and Austin, 2001). Reactions on faults differ with either a “fail stop” or with distribution message about the fault across the network (Ali et al. 2007).

Intermittent faults appear occasionally and their appearance is linked to the stressful operating conditions. Examples of this type of fault include power supply voltage noise or timing faults due to inadequate cooling systems and data dependent design errors. These execution errors are the most difficult to find (Weaver and Austin, 2001). For this reason and due to their rare occurrences not much attention is paid to intermittent faults.

From the system’s viewpoint all faults corrupt and loose data at different levels: single bit, several bits or burst of bits. For networks typical faults are:
- **Line outages**, in which a circuit fails;
- **White noise**, caused by thermal energy;
- **Impulse noise**, burst errors such as lightning, power surges, and poor connections;
- **Cross-talk**, one circuit picks up signals from another adjacent circuit;
- **Echoes**, poor connections causing the signal to be reflected back to the transmitting source;
- **Attenuation**, signal weakening over distance;
- **Jitter**, caused by small variations in the amplitude, frequency, and phase of a signal;
- **Harmonic distortion**, incorrect amplification of an input signal.
Undoubtedly, faults need to be tolerated using mechanisms for fault detection and recovery, i.e. using GAFT.

3.2. GAFT implementation for the router

The traditional process of monitoring faults as an algorithm is generalized further using (Avizienis 1971) and (DeAngelis and Lauro, 1976) as a starting point. An extention of the algorithm of fault tolerance initially introduced in (Schagaev 1986) was further developed and became known as generalized and algorithm of fault tolerance (GAFT) (Schagaev and Zalewski, 2001).

The primary function of fault monitoring assumes a sequence of steps, such as fault detection, fault type identification, faulty component(s) location, and hardware reconfiguration, to achieve a repairable state and re-establishment of a correct state(s) for the system and user software.

GAFT steps are implementable using various redundancy types; can be performed either synchronously or asynchronously using hardware or software redundancy. Some examples of hardware (HW) and software (SW) redundancy are:

- HW(I) - a hardware redundancy to keep extra information for GAFT purposes such as redundant line or 1-bit register of data to check errors of data;
- HW(T) - synchronously applied hardware redundancy such as special hardware delay (latch) to avoid malfunctions caused by racing of signals;
- SW(S) - asynchronously applied software redundancy such as periodic hardware testing procedures performed in spare time;
- SW(I) - informational redundancy of the program deliberately applied to recover a system - usually known as recovery points (DeAngelis 1976, Schagaev 1986).

The efficiency of hardware and system software implementation of fault tolerance differs in terms of type and power of faults, the variety of faults tolerated and on performance degradation. The GAFT is initiated by external reasons with the function of detecting faults.

The physics of the fault itself have to be presented somehow in the structure of the algorithm, as recovery procedures from malfunction and permanent damage of hardware are different. Due to substantially higher rate of occurrences GAFT should deal as soon as possible with hardware malfunctions. Above all they are easier to recover. The closer the fault type detection is to the beginning of the algorithm of fault tolerance, the faster the algorithm completes its function in the case of malfunction. Taking into account that the ratio of malfunctions to permanent faults is roughly 10^{-3} a differentiation of types of fault will obviously increase reliability. The Generalised Algorithm of Fault Tolerance as applied to router operation is shown in Figure 2.

Incomplete checking and uncertainty in the pattern of faulty behaviour are attributed to latency of hardware faults. In this case even pre-arranged recovery points (RP) created during program execution might be damaged, as they may already contain erroneous data, therefore even several steps of recovery would not be enough to achieve a correct (at least consistent) state of hardware and software to continue the execution.

![Fig. 2: GAFT for router](image)

A special phase of GAFT is introduced to locate the correct recovery points (RP) and further recover a state of the system from the trusted recovery point (Schagaev 1986, 90). Once is proved that the software state was not damaged or that the recovery was successfully completed, the router operational functioning is resumed. Both checking and recovery steps and their implementations are important and unavoidable elements of GAFT. Recovery points should be made during the regular operation of a router and should also be monitored by the system software.

(Schagaev, 2008) presents detailed analysis of malfunction impact on the reliability of hardware. However, in order to support the statements made here let’s consider a system without redundancy where:

- the rate of permanent fault for the system is: \(\lambda_{pf1}\)
- the malfunction rate of the system is: \(k \lambda_{pf1}\)

The probability of operation without failure within time interval \([0,T]\) is determined by the formula:

\[
P(t) = e^{-(1+k) \lambda_{pf1} t}
\]  

(1)

Considering that for modern technologies \(k\) varies from 10^{3} to 10^{6} (the latter applies to aeronautics) \(t\) his simple observation shows that reduction of malfunction impact increases reliability by one order of magnitude. Note that this statistical evaluation of reliability includes two different processes (malfunctions and permanent fault appearance) which are considered as two independent random processes.

Fault tolerance can be implemented or supported using different levels of time or structural redundancies within hardware and system software. System software support for fault tolerance assumes several levels of implementation. These levels are instruction level, procedure level, module level and task level (Schagaev, 2008), as shown in Figure 3.
Taking into account that the router hardware is highly unlikely to be modified in the foreseeable future, levels of procedure and module recoverability, if implemented in system software, may be the only option available towards increasing reliability. In turn, checking the various processes of router hardware consistency is possible to implement using the approach described by (Kaegi-Trachsel et al. 2009).

3.3. Algorithm for FT routing table lookup

From a reliability point of view router hardware consists of three units (or segments) and forms a sequential structure with input, routing and output processing segments. Failure of any of these leads to complete failure of the router.

From the system software point of view the three processes which form the core of the router operating system are the inbound, routing and outbound processes. As it was proposed by (Schagaev, 1990), the processes of checking, recovery point formation and recovery point activation have to be implemented within the router operating system for all three router hardware segments.

When a packet arrives at a router’s inbound interface it is transferred to the processor so that changes in its header can be made prior to it being forwarded towards its destination. During this process the packet can be corrupted whilst being read from the input buffer, written into the processor cache, processed or written into output buffers, as shown in Figure 4. If the integrity of the packet is proven then it will be processed further and then is sent out to the outbound buffers.

In the case of a faulty packet several delays might take place which can affect the reliability and performance of the whole network due to the packet not being read, processed or rewritten properly at various stages of the routing process.

The proposed solution involves the implementation of GAFT and the following steps added to the routing process:

- A coding technique employed to check if the packet has been read correctly during its transfer from the input buffer to the Local Processing Subsystem (LPS) and from the LPS to the output buffers.

- Checking of the router hardware consistency to be implemented using an approach such as that described by (Kaegi-Trachsel 2009).

Fig. 3: Levels of GAFT implementation (Schagaev, 2008)

Fig. 4: Architecture of network interface of router with distributed processing

- Recovery points (RP) have to be established for inbound and outbound processing, and routing processing.

When necessary the router operating system initiates recovery procedures using a family of established recovery points - different for all three router hardware segments: \{RP$_i$\}, \{RP$_o$\}, \{RP$_sw$\} (the indexes stand for input, output and system software RP respectively).

Structural redundancy in the router system software or hardware might be used to detect and recover processing faults. Two options are possible:

- Using as system software redundancy an implementation of special algorithms - SW(dS), (dS stands for redundancy less than 100%). SW checks whether the resulting processed packet is consistent, and establishes the above mentioned family of recovery points; or,

- Using hardware redundancy, such as a two processor configuration - (redundancy of this configuration is HW(2S)). Parallel use of hardware with matching results has exceptional coverage of faults and virtually zero-time overheads (Sogomonyan et al. 1990). For duplicated router hardware configuration further processing of the result to the output buffer takes place only if both results from processing the packet match.

An interesting option of system software - hardware (SSW-HW) support of router fault tolerance might be observed from Figure 4. When a fault is detected and located within inbound hardware the rest of router architecture can continue regular processing, while inbound segment is repaired. The same is true for the outbound hardware segment.

An even stronger argument for the use of SSW-HW implementation of router fault tolerance lays in the ability of using a Local Processing Subsystem (LPS) to handle any “hicups” of the central route processor and vice versa – the central route processor can handle any malfunctions of the LPS. This approach virtually excludes the core of the router as far as reliability is concerned.

The process of making a router fault tolerant is shown as an algorithm in Figure 5. When a packet arrives at the input buffer checking and recovery information will be generated. If the integrity of the packet content is proven then the pack-
et will be processed; otherwise reading of the packet from the buffer will be repeated an arbitrary number of times, denoted as $m$. This action filters the malfunction impact on the incoming segment of the router hardware. In case of a permanent fault respected steps of GAFT are performed, making the router a fail-safe device. A similar procedure is applied for output packet processing. Dealing with routing of a packet with proven consistency includes several steps (6…9) of the algorithm as shown in Figure 5.

Again, in the case of any detected inconsistencies, a procedure or re-reading the packet will be applied with $n$ or less (if successful) number of iterations. Finally, for the router outbound segment, together with automatic formation of recovery points (mentioned as redundant information generation), a process of checking and repetition is implemented.

Note that these two processes have a semantic difference: the checking and formation of recovery points is synchronous and is performed constantly along the routing process. In turn, recovery actions and repetitions of reading from caches is asynchronous and activated only when packet integrity is detected. These sequences include operators 4, 5, 7, 8, 9, and 13, 14.

Due to any physical difference of the intensity of use for input, system and output hardware areas of router number of repetitions for filtering the malfunctions might be different, denoted as $m$, $n$, $l$. When a permanent fault in the router is detected a GAFT segment for permanent faults is initiated (box 15).

4. CONCLUSIONS

The fault tolerance of routers and ways to improve their reliability were discussed. It has been shown that absence of consistency in the proof of a condition of router hardware is the major drawback in improving reliability. To cope with this problem an application of the generalized algorithm of fault tolerance (GAFT) was proposed. This algorithm of router fault tolerance was presented here in detail. Various redundancy types have been shown to be applicable for different steps of the algorithm making an option to design flexible real-time fault tolerant systems. The implementation of an algorithm assumes support and coordination of the process of hardware checking and formation of three sets of recovery points for inbound, routing and outbound hardware of the router respectively.

---

**Fig. 5:** Algorithm for fault tolerant routing table lookup process with the concept of recovery points
Recovery procedures include searching for the correct recovery point to restart the router operation. The probability of this procedure depends on the quality and consistency of the checking procedures. Recovery actions might be implemented in different router hardware segments thus reducing performance degradation of the router as a whole even during the recovery process.

The distributed processing architecture of the router (central route processing and local processing subsystem) enables mutual checking and recovery procedures to be performed and excludes the core of the router in terms of reliability.

REFERENCES