Fault-Models in Wireless Communication: Towards Survivable Wireless Networks*

Axel W. Krings
University of Idaho
Moscow, Idaho 83844-1010, USA
krings@uidaho.edu

Abstract

This research introduces a new approach to modeling wireless network reliability under diverse fault assumptions. It allows for quantifying reliability and offers potential for modeling survivability. The general model is presented as a join graph of cliques, that allows for horizontal and orthogonal cross-monitoring. This allows for the determination of the maximal potential fault tolerance. The two-dimensional cross-monitoring approach is related to recent research addressing omission faults [5]. Finally an example of its use is given in which we consider benign and omission faults and utilize primary-backup scheduling, specifically backup-backup link scheduling, as fault tolerant mechanisms.

1 Introduction and Background

Wireless applications have experienced tremendous growth in recent years. Especially in the area of ad hoc and sensor networks there are many new challenges due to their features and the inherent characteristics of wireless technology. Ad hoc and sensor networks operate in environments where the restrictions on nodes with respect to their computation and communication capabilities vary greatly. The characteristic property of these networks is the dynamic nature of computation and communication, may it be as the result of limited battery power of the nodes or due to their physical movement, to name a few. The reliability of wireless networks has been addressed primarily in the context of quality of service (QoS). The main considerations have been routing and the overhead resulting from dealing with disruptions of the communication paths. However, due to the nature of wireless communication, the network model also raises many security related concerns. Nevertheless, the same feature, i.e., wireless broadcast, which creates security problems, can also be part of the solution in addressing diverse faults.

Much research has considered routing issues, which present great challenges in the rather dynamic environment, and many protocols have been introduced. However, most research has focused on operation in benign environments, and security considerations were not the driving motivation.

This research takes a step back from specific implementation-driven approaches and considers what the implications of the wireless network on the fault models are and vice versa. At the basis are the fundamental assumptions associated with fault models used in the reliability community.

Network Representation: Before discussing reliability and survivability issues of wireless systems in the context of fault models, the network needs to be abstracted. A wireless network will be represented as a digraph $G = (V, E)$, where computational nodes are the vertices and communication links are the edges. Specifically, given two nodes A and B, represented by v_a and v_b respectively, then if B can receive the signal of A, edge e_{ab} is in E. Similarly, if A can receive the signal from node B, edge e_{ba} is in E. The choice of a digraph over an

*This work has been supported by by a grant from the INL (Idaho National Laboratories).

undirected graph stems from the general philosophy of cross-monitoring, which will be the basis of fault detection mechanisms presented later.

Next, we want to define several fundamental graph operations and properties. Given two graphs G_1 and G_2 with vertex sets V_1 and V_2 and edge sets E_1 and E_2 respectively, the *union* $G = G_1 \cup G_2$ has $V = V_1 \cup V_2$ and $E = E_1 \cup E_2$. Their *join*, denoted by $G_1 + G_2$, consists of $G_1 \cup G_2$ and all edges joining V_1 and V_2. Finally, a *clique* is a fully connected subgraph of G.

Fault Models: Fault models have played a major role in reliability analysis and in agreement and consensus algorithms. Many different types of faults have been defined, some having orthogonal properties [2]. For example, fail-stop behavior implies that the faulty processor ceases operation and alerts other processors of this fault. Crash faults, on the other hand, assume that the system fails and looses all of its internal state, e.g. the processor is simply down. One speaks of omission faults when values are not delivered or sent, e.g., due to a communication problem. If outputs are produced in an untimely fashion, then one speaks of a timing fault. Transient faults imply temporary faults, e.g. glitches, with fault free behavior thereafter. If transient faults occur frequently, one speaks of intermittent faults. This set of fault types is by no means complete and serves only as a basic introduction. The definition of faults seems to change with the application domain. For instance, fault models suitable for computer dependability may not necessarily match the behavior of network and computer security applications [2].

Whereas the previous paragraph considers different types of classical faults, their behavior with respect to other processors can be described in simpler models which have been used with in replication and agreement algorithms. Specifically, fault models have been considered whose main behavior types are *benign*, i.e., globally diagnosable, *symmetric*, i.e., faulty values are seen equal by all non-fault processes, and *asymmetric* or malicious, i.e., there are no assumptions on the fault behavior.

Within the context of communication models assumed in this research we want to use the five fault hybrid fault model of [3], which extended the three fault model of [6] to a five fault model by considering transmissive and omissive versions of symmetric and asymmetric faults respectively.

Redundancy: In order to tolerate a fault by recovering the faulty information, several redundancy mechanisms have been used. *Time Redundancy* addresses that certain actions are performed several times, skewed in time, and that some majority measure is used. It is often used for redundant sensor readings in embedded systems. *Information Redundancy* uses redundant information, e.g., extra bits, to reconstruct lost information. Error correction codes are a typical example. *Spatial Redundancy* assumes that redundant units, e.g., processors or communication links, are available. Failed units are masked by the redundant units. For example, if one considers b benign and s symmetric faults, then one needs $N > 2s + b$ redundant units for masking the effects of the faults.

One interesting observation is that in wireless systems there is only limited opportunity for asymmetric faults. Specifically, transmissive asymmetric faults are in general not possible within one broadcast domain, since all nodes within the range of the sender receive the same information. However, there is potential for asymmetric faults when messages traverse over disjoint paths.

Fault Assumptions: It should be pointed out that faults are seen only in the context of their definition in the specific fault models under consideration. Standard mechanisms that address reliability or security concerns, e.g., authentication, are "tools" that have impact on the fault types that can be produced. For example, a fault that is detected by the authentication mechanisms is a benign fault. If the authentication method fails to expose the malicious act, e.g., a method was found to circumvent the authentication mechanism, then this fault has the potential to be symmetric or asymmetric. There are many approaches that utilize tools from the field of security and fault-tolerance in order to increase security and reliability however, in the end their impact on the fault is what really counts. The mechanisms have the potential to lessen the severity of the fault, e.g., being able to downgrade the possible fault from symmetric to benign. Our goal is to derive a general reliability model that can then be used to aid in the decision process on which mechanisms are feasible and what the impacts are with respect to reliability. This model assumes the philosophy of a general model to expose the theoretical limitations and possibilities.

2 Network Model

The network model is defined next, starting with the relationship between the wireless network and the formal representation as a flow graph.

The network is represented by digraph $G = (V, E)$. The left part of Figure 1 shows a sample network consisting of 4 wireless nodes. The broadcast area is indicated for each node by ovals. The broadcast area of node 1 is shown shaded, the other areas are not. Overlapping areas imply a communication path between the nodes only if the receivers of the nodes are in the broadcast area of the neighboring nodes. As can be seen, node 2 can receive from node 1 and vice versa. Node 3, however, can only receive from node 1, but its broadcast area does not reach another antenna. Lastly, even though the broadcast area of node 1 and 4 overlap, neither antennas can receive each other's signal. The graph on the right-hand side shows the network digraph G, implementing a reachability graph where an edge e_{xy} is present only if node x can receive the signal of node y.

Graph G is conceptually related to a flow graph of a network. For wired networks the flow of packets follows a specific path in the graph, each packet traversing a specific link. Thus, the flow at a node with multiple outgoing edges will utilize exactly one edge for a packet.

In wireless networks this is different. Due to the broadcast nature of wireless communication a packet always "traverses" over *all* outgoing edges of a node, i.e., any node within the broadcast domain can see the message.

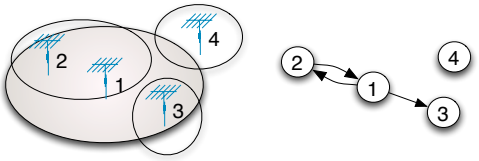

Figure 1: Wireless Network and Graph G

Two Dimensions of Cross-monitoring:
Before describing the network model in detail, we need to address the difference between fault detection and fault correction capabilities. By the definition of benign faults, these kinds of faults are trivial to detect. However, other faults, e.g., omissions, may only be detected by external mechanisms such as timeout mechanisms or cross-monitoring [5]. A timeout constitutes an omission fault that exhibits benign behavior. However, relying on timeout mechanisms to detect omissions is expensive. The reason is that the timer values are usually very conservative, since otherwise there is the potential for excessive timeouts.

The basic mechanism for fault detection and consequent potential fault correction will be cross-monitoring. In general, every monitor node v_m has the potential to cross-monitor any node v_s if graph G contains edge e_{sm}. A prerequisite for effective cross-monitoring is however that there is a reference that can be monitored against. The monitor node needs to have the packet or some signature of the packet to check against. This prerequisite has important implications on the queue sizes of nodes and realities of cross-monitoring.

Cross-monitoring in the direction of the communication path will be referred to as *horizontal cross-monitoring*. It can expose corruption and omissions but cannot verify actual delivery. The watchdog monitoring scheme presented in [5] constitutes horizontal cross-monitoring, wheras monitoring is limited to the principal communication path.

On the other hand, in topologies allowing for multi-path communication, cross-monitoring can also be orthogonal to the communication path. This dimension of monitoring will be called *orthogonal cross-monitoring*. It can be shown that, in general, horizontal monitoring has the potential to detect faults, and that orthogonal monitoring can detect and possibly correct faults, depending on the fault types assumed.

General Graph Model:
We will now define the general graph model as a two-dimensional model, featuring a horizontal and orthogonal plain. For two communicating nodes v_S and v_D a join graph will be derived from the wireless infrastructure graph. Let G' denote the infrastructure graph. Now construct G as the network graph between source v_S and destination v_D as follows: (1) A path between v_S and v_D defines the primary communication path. (2) Let C_1 be a clique of all vertices v_i that are incident from v_S, i.e., for each $v_i \in C_1$ there exists e_{Si}. (3) For each v_j in the primary communication path define C_j as a clique of *all* vertices v_i, for which there exists an edge e_{hi} from *all*

$v_h \in C_{j-1}$. (4) Let C_D be a trivial clique containing only v_D. Figure 2 shows the general structure of G. Note that each shaded oval is a clique containing one node of the principal communication path. Furthermore, by the construction of the graph, there is an edge from each vertex in C_i to each vertex in C_j. This makes the combined subgraph $C_i \cup C_j$ a join graph. Also note that, if all edges between C_i and C_j are bidirectional, then $C_i \cup C_j$ forms again a clique.

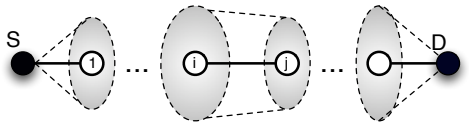

Figure 2: General Join Graph

Fault-toleranc:
Given the general joint graph, one can determine the fault-tolerance of the communication between the source and destination. In the context of our model there are several principal approaches to recovery.

First, detection can be used to re-request a packet, as is the case in TCP. Lost or corrupt packets, detected by various mechanisms such as CRC, timeout or horizontal cross-monitoring, are re-requested by the transport layer. This essentially mimics timing redundancy, where b benign faults require a total of $b + 1$ transmissions.

Second, cross-monitoring based on comparison of duplicated packets constitutes spacial redundancy. As such, it is burdened with the high cost of replication. In general, packet duplication on k disjoint paths can tolerate $b = k - 1$ benign faults, or $s = \lfloor (k-1)/2 \rfloor$ symmetric faults.

In order to determine the reliability of a communication implemented as a join graph we will utilize the concept of Reliability Block Diagrams. Specifically, the graph is a series graph, where each component is in turn a *parallel*, i.e., 1-of-N construct, or a k-of-N construct. In reference to Figure 2, the graph is a series of constructs, representing the cliques, i.e., $v_S, C_1, ..., C_i, C_j, ..., v_D$. If only benign faults are considered, the reliability $R_i(t)$ of a the construct representing C_i, consisting of N_i nodes, is determined by $R_i(t) = 1 - \prod_1^{Ni}(1 - R(t))$. R(t) is the reliability of a node and is assumed as $R(t) = e^{-\lambda t}$, where λ is the fail rate. Note, that this definition of reliability is very limited and arguably non-suitable for modelling malicious human act.

3 Applications

The above concept has been used to model several approaches to fault-tolerance in wireless networks. First, we were able to show the power and limitation of horizontal cross-monitoring as shown in [5], where only the principal communication path was exercised. However, a more formal reliability analysis can be possible. Specifically, we suggest the use of standard reliability models, rather than their measure for their so-called pathrater. Second, we could show that one can eliminate much of the overhead associated with redundancy by adapting the notion of primary-backup scheduling, which was introduced in the context of fault-tolerant scheduling in real-time multiprocessor systems, including [1, 4]. The results from this research are currently in preparation for submission.

References

[1] R. Al-Omari, et.al., Efficient overloading techniques for primary-backup scheduling in real-time systems, *J. Parallel and Distributed Computing*, Vol. 64, Issue 5, pp. 629648, May 2004.

[2] A. Avizienis, J.C. Laprie and B. Randell, Fundamental Concepts of Dependability, *Information Survivability Workshop (ISW-2000)*, Boston, Oct. 24-26, 2000.

[3] M.H. Azadmanesh, and R.M. Kieckhafer, Exploiting Omissive Faults in Synchronous Approximate Agreement, *IEEE Trans. Computers*, 49(10), pp. 1031-1042, Oct. 2000.

[4] S. Ghosh, et.al., Fault-Tolerant Scheduling on a Hard Real-Time Multiprocessor System, *Proc. Intl Parallel Processing Symposium*, pp. 775782, 1994.

[5] S. Marti, et.al., Mitigating routing misbehavior in mobile ad hoc networks, *Mobile Computing and Networking*, pp. 255-265, 2000.

[6] P. Thambidurai, and Y.-K. Park, Interactive Consistency with Multiple Failure Modes, *Proc. 7th Symp. on Reliable Distributed Systems*, Oct. 1988, 93-100.