Fault-Models in Wireless Communication: Towards Survivable Ad Hoc Networks

Axel W. Krings and Zhanshan Ma
University of Idaho
Moscow, ID 83844-1010, USA

***Abstract*— A new approach to modeling wireless networks is presented that allows for the determination of network reliability using diverse fault assumptions. It is shown that one can exploit network topologies by taking advantage of the broadcast paradigm of wireless communication to detect and possibly correct benign and malicious act. Specifically, a general wireless network model is presented that maps subsets of the network to join graphs of cliques. This join graph allows for horizontal and orthogonal cross-monitoring, which exposes the theoretical limitations of fault detection and correction. For ad hoc and sensor networks the two-dimensional cross-monitoring scheme offers great flexibility with respect to establishing topologies capable of meeting reliability and survivability requirements. Recent approaches addressing tolerance to "misbehaving" nodes are shown to be special cases of the general model.**

***Index Terms*— Wireless network, ad hoc network, sensor network, security, survivability, reliability.**

I. Introduction and Background

With the tremendous growth of wireless applications in recent years comes great concern for the lack of reliability, security and survivability [28]. Especially in military applications 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 [27]. 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, e.g., may it be as the result of limited battery power of the nodes or due to their physical movement. 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. As a result, many protocols have been introduced. However, in critical applications operating in hostile environments the security and survivability requirements may be much higher than usual and fault assumptions should include pathological behavior. Furthermore, most research has focused on operation in benign environments where security considerations were not the driving motivation. Nevertheless, the same feature, i.e., wireless broadcast, which is at the source of security problems, can also be part of the solution in addressing diverse faults.

This work has been supported by an LDRD grant from the Idaho National Laboratory (INL).

This research takes a step backwards 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.

A. Fault Models

The easiest assumptions about faults is that they exhibit fail-stop behavior, which implies that the faulty processor ceases operation and alerts other processors of this fault. However, there are more realistic definitions of faults. For example, crash faults assume that the system fails and loses 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. The list goes on and the diversity of faults has been the primary motivator for the definition of 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 proposed ranging from those defined in hybrid fault-models, e.g., [30], [2], to those considering issues related to security [1].

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 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 [30].

Within the context of communication models assumed in this research we subscribe to the five-fault hybrid fault-model of [2], which extended the three fault-model of [30] by considering transmissive and omissive versions of symmetric and asymmetric faults. Specifically, the fault types in are:

1) Benign: a benign fault is self-evident to all nodes.
2) Transmissive symmetric: a single erroneous message is delivered to all receiving nodes. The messages, even faulty, are all identical. This kind of fault captures the meaning of symmetric faults in [30].
3) Omissive symmetric: no message is delivered to any receiving node. As before, all nodes are affected the

same, however, the omissive behavior results in the destination nodes to most likely take different action as if the message had been received.

4) Transmissive asymmetric: this fault can exhibit any form of arbitrary asymmetric behavior, capable of delivering different erroneous messages to different receivers. This interpretation captures the meaning of asymmetric faults in [10], [30].
5) Strictly omissive asymmetric: a correct message is delivered to some nodes and no message is received by other nodes. Here, the omissions have the capability of affecting the system in an asymmetric way, since those nodes who have not received the message will most likely react differently, e.g. selecting a default action, to those who have received the message.

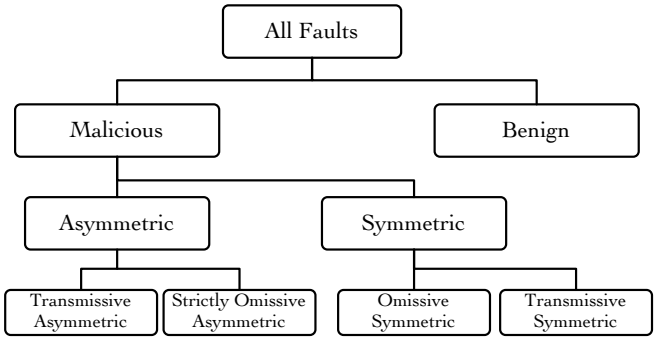

Fig. 1. Overview of Fault Models [2]

Figure 1 shows the relationship between the different fault-models. The root represents the fault-model shown in [10] which assumes that all faults are malicious, i.e. asymmetric. The model on the next level considers that assuming that all faults are malicious is perhaps too conservative and thus partitions the fault into asymmetric and benign faults [19]. Asymmetric faults are furthermore partitioned on the next level [30], and at the lowest level a further partitioning into omissive and transmissive behavior is assumed [2]. Note that the definition of benign faults stays constant for all hybrid fault-models.

B. 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. Time redundancy is often used for redundant sensor readings and is frequently used in embedded systems. *Information Redundancy* uses redundant information, e.g., extra bits, to reconstruct lost information. Error correction codes are a prime example of this redundancy type. *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. There is, however, potential for asymmetric faults when messages traverse over disjoint paths.

C. 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 authentication fails to expose 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 faults they can produce 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.

This work does not focus on approaches that help increase security attributes via mechanisms such as encryption or authentication, nor does it focus on specific transport or data-link handshakes that increase reliability or QoS in general. It assumes that all such mechanisms are used according to the application's mission. Our goal is to derive a general reliability model that can then be used to aid in the decision process on what mechanisms are feasible and what the impacts are with respect to reliability. This model assumes the philosophy of deriving a general model to expose the theoretical limitations and possibilities. It is more in line with the approaches taken in [23] or [24] which present models for fail-stop processors and secure agent systems respectively.

D. Related Work

Since this work relates to tolerance of faults of different types under possibly pathological scenarios, we need to explore redundancy mechanisms. As such, any approach utilizing multipath and multiflow communication could have the potential for tolerating faults, if these concepts are exploited for reliability [21]. Many multipath and multiflow approaches have been presented in the literature, but their focus has not been on tolerating diverse faults but have rather been limited to overcome benign link or node faults. For example, the concept of multiflow has been used in [31] in the context of QoS enhancement, however, the focus is on transmission congestion. Multipath routing has been used to increase end-to-end reliability, e.g., the MP-DSR protocol in [13] forwards outgoing packets along multiple paths that are subject to a particular end-to-end reliability requirement, but the impact of faults as described here are not considered.

An approach actually considering the impact of topology was shown in [14] where communication topology optimization is treated as a linear programing problem. However, there

is no spatial information redundancy involved. The impact of eavesdropping is considered in [12], where a secret sharing approach is used. Whereas this addresses confidentiality issues, it does not address tolerance of a fault. In fact, more general data distribution schemes and their impact on survivability have been extensively studied within the PASIS project at CMU [32] and their suitability to agent systems have been shown in [8].

An on-demand routing scheme called Split Multipath Routing (SMR) was shown in [11]. The protocol establishes and utilizes multiple routes of maximally disjoint paths to minimize route recovery and control message overhead. Again omission faults are considered. Predicting fault behavior has been advocated in [29], however this is extremely difficult even in the case of link failures for malicious act. Similarly, intrusion detection may be unrealistic due to the excessive resource constraints associated with information required by the IDS [20].

Primary and backup communication paths are considered in [16]. However, disjoint paths are not exploited for data redundancy but discarded as unwanted overhead. In their use of redundant disjoint paths the overhead to resilience tradeoff becomes unfavorable for a larger number of paths [5], [17]. Rather than banking on multiple paths, robustness to node failures is addressed in [33] by using the concept of reliable nodes and reliable paths. Whereas robustness is significantly increased, the gain is due to restrictions on faults of the reliable nodes.

An approach actually addressing fault-tolerance was presented in [18] where "misbehaving" nodes causing omission faults were detected by so-called "watchdogs". The impact of nodes that failed to relay packets was shown and a method was presented that allows for tolerance of such nodes. The concept was extended in [22] where collaborating groups of malicious nodes were considered. In [3] the effectiveness of various watchdog schemes was investigated. Their results suggest that watchdog schemes are indeed able to detect a number of attacks such as omissions and certain symmetric faults but exposes limitations, e.g., fabrication of false route error messages. Wormhole attacks were addressed in [26], where statistical analysis was used for detection of nodes which launch them. Detection of malicious behavior due to observation of monitoring nodes operating in promiscuous mode was shown in [4].

The watchdog and monitoring strategies above will be combined below and expanded in a unified network model. This model will expose and overcome the limitations of the "horizontal" watchdog approach by introducing orthogonal mechanisms.

II. Network Model

We now define the network model starting with the relationship between the wireless network and the formal representation as a flow graph.

A. Network Graph G

The foundation of the network model is the abstraction of its infrastructure. A network will be represented as a digraph $G = (V, E)$, where computational nodes are the vertices and communication "links" are the edges. The left part of Figure 2 shows a sample network consisting of 4 wireless nodes, where broadcast areas are indicated for each node by ovals. Note that some antennas have circular broadcast patterns whereas others are narrow or directional. The broadcast area of node 1 is shown shaded. 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, whose antenna is rather directional, can receive from node 1 and vice versa. However, node 3 can only receive from node 1, as its broadcast area does not reach another antenna. Lastly, even though the broadcast area of nodes 1 and 4 overlap, neither antennas can receive each other's signal. The graph on the right-hand side shows the associated network digraph G, implementing a reachability graph. In general, given two nodes represented by vertices v_i and v_j in V, edge e_{ij} is in E only if node j can receive the signal of node i.

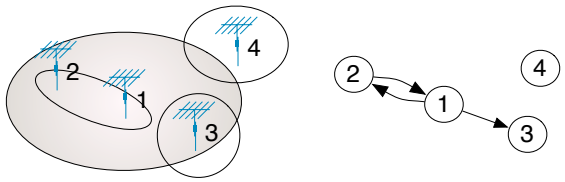

Fig. 2. Wireless Network and Graph G

Next, we want to define several fundamental graph operations and properties. Given two graphs G_i and G_j with respective vertex sets V_i and V_j and edge sets E_i and E_j, the *union* $G = G_i \cup G_j$ has $V = V_i \cup V_j$ and $E = E_i \cup E_j$. The *join* $G = G_i + G_j$ is shown in the example of Figure 3. $G = G_i + G_j$ consists of $G_i \cup G_j$ and all edges joining V_i and V_j, i.e., $\forall v_p \in V_i, v_q \in V_j, e_{p,q} \in E$. Finally, a *clique* is a fully connected subgraph of G.

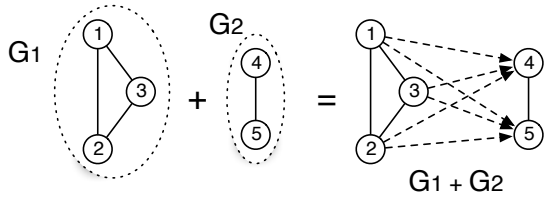

Fig. 3. Join Operation (+) of two Graphs

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. If the network is composed of fixed links and wireless links, then colors could be

assigned to edges to differentiate between broadcast/multicast and point-to-point communication.

B. Cross-monitoring

Before describing the network model in detail, we need to address the difference between fault detection and fault correction capabilities, which are fundamental to the interpretation of the model. By the definition of benign fault, this kind of faults are trivial to detect. However, other faults, e.g., omissions, may only be detected by external mechanisms, e.g., timeout mechanisms or cross-monitoring [3], [4], [18], [22]. At best, a timeout constitutes an omission fault that exhibits benign behavior. However, relying on timeout mechanisms to detect omissions is expensive since the timer values are very conservative, e.g., the Retransmission Timeout period (RTO) in TCP is measured in seconds. In general, choosing the value too small has the potential for excessive timeouts. Moreover, omissions have the potential to generate strictly omissive asymmetric faults in sensor networks.

The basic mechanism for fault detection and consequent potential fault correction will be cross-monitoring. Cross-monitoring has been used extensively in reliable system design for decades, e.g., Space Shuttle [25] or MAFT [7]. We consider cross-monitoring of data packets in wireless communication. 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 thus on the realities of cross-monitoring.

Figure 4 explains cross-monitoring and potential detection and correction capabilities. Consider the routing path from source S to destination D in the network shown in the top part of the figure where the relative placement of vertices reflects the physical location of the nodes. The undirected edges along the routing path indicate bidirectional communication, the dashed edges indicate links capable of cross-monitoring[1]. Only nodes that will be referred to later are labeled. The placement of the vertices in the graph relate to the physical position of the nodes.

The bottom of Figure 4 shows the logical graph, where vertices that cannot contribute to cross-monitoring have been suppressed. Let us denote the physical and logical graphs by G^P and G^L respectively. Consider node v_S in G^P. All v_j incident from v_S can receive the packet. Node v_1 can see the packet, but is not capable of cross-monitoring any other node. Node v_S can confirm that the packet was received by v_3 and can itself cross-monitor if the packet was forwarded to v_4. This was shown in [3], [4], [18] and [22], where the monitor was called watchdog. However, since v_S cannot see v_4, it can only observe if v_3 fails to forward or falsifies the packet. Even if v_3 appears to forward the packet correctly, v_S has no immediate way of knowing if v_4 actually received it. These limitations

[1]There is no difference between an undirected edge and an edge with two arrow heads. We simply omitted the heads to avoid visual clutter.

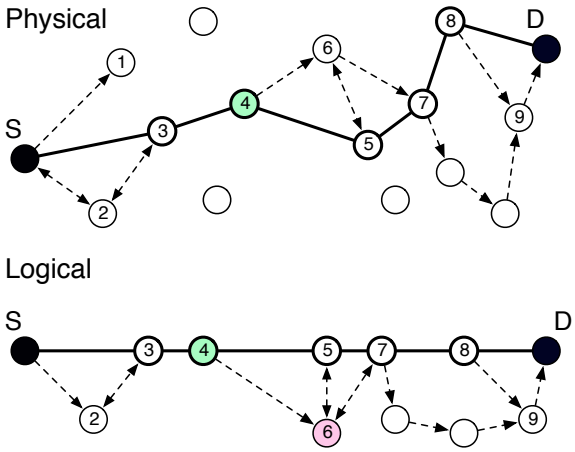

Fig. 4. Cross-monitoring in a Network

were pointed out in the works previously cited. Node v_2 adds no value in overcoming these problems and can only be used as an alternate route in case e_{S3} would fail.

Next, consider node v_4 in G^L, whose packet sent to v_5 is also seen by v_6. Nodes v_4 and v_6 can verify that v_5 received and forwarded the packet. However, only v_6 can verify if v_7 actually received it. Thus, in the case of a strictly omissive asymmetric fault, e.g., v_5 does not forward the packet to v_7, then v_6 can detect the omission and supply the packet.

In all cases of cross-monitoring it is required that the packet is present in the monitor and the target node. Assume the case of v_7 in G^L who forwards the packet towards v_D. The packet could be forwarded via v_8 or using the lower path containing v_9. Due to the different hop counts in the upper and lower path, the packet may arrive in v_8 and v_9 at different times. In order to be able to cross-monitor, the packet would have to be in v_9 when v_8 sends it to the final destination. This however may put unrealistic constraints on queuing buffer sizes.

C. Two Dimensions of Cross-monitoring

The previous subsection exposed that cross-monitoring can occur in the direction of the network traffic, e.g., in Figure 4 v_S could be used to cross-monitor the packet forwarded by v_3 to v_4. This cross-monitoring will be referred to as *horizontal cross-monitoring*. It can expose corruption and omissions but cannot verify actual delivery, nor can it detect pretentious forwards to non-existing bogus nodes [3], [18], [22]. The watchdog monitoring scheme constitutes horizontal cross-monitoring. More precisely, their monitoring is limited to the *principal communication path*.

On the other hand, it was shown above that cross-monitoring could also be orthogonal to the principal communication path, e.g., v_6 could cross-monitor v_5 to ensure that the packet from v_4 was forwarded, and v_7 to confirm delivery via its acknowledge to v_5. This dimension of monitoring will be called *orthogonal cross-monitoring*. An approach using limited orthogonal cross-monitoring based on counting of incoming and outgoing packets was shown in [4]. We will show 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 type that is assumed. The resulting model exceeds the capabilities of any cited research.

D. 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 infrastructure graph. Let G' denote the infrastructure graph.

General Join Graph: 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 principal 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 edge e_{Si}.
3) For each v_j in the principal 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.

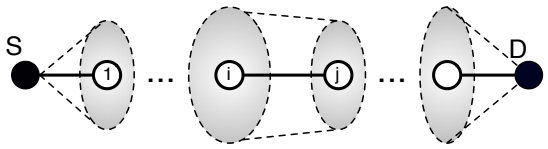

Fig. 5. General Join Graph

Figure 5 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 6 shows a hypothetical join graph G that could have resulted from the physical graph shown in Figure 4 if one were to increase broadcast power or make minor node rearrangements. In the context of [18], where only horizontal monitoring is possible, only omissive faults along the principal communication path can be detected. In their case, this was then used to determine alternative path in their so-called *pathrater*. However, recall that the approach did not detect non-delivered packet forwards or malicious behavior like pretentious forwarding to non-existing bogus nodes. Cross-

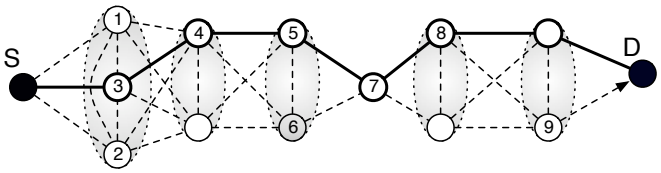

Fig. 6. Join Graph Example

monitoring using join graphs overcomes these limitations and offers the potential to react to the observed behavior. As will be pointed out next, there is however a requirement that redundant packets overlap in the queues of the participating nodes.

E. Cross-monitoring Cost

In general, there is a temporal and space dimension associated with cross-monitoring. Temporality relates to the fact that cross-monitoring can only be performed as long as the packet to be monitored is still in the queue. Once the packet leaves the queue, there is no frame of reference for monitoring. This puts a temporal constraint on the cross-monitoring nodes, i.e., the packet on any participating nodes must have temporal overlap in the respective queues. Obviously, as the difference in propagation delay between two packets to be monitored grows, so must the queues of the participating monitors. In the general model this is addressed by limiting cross-monitoring to a general join graph. If one allows for more general graph models, then issues of larger variation in the overlap time need to be considered. An example of this would be the establishment of a slow communication link between v_7 and v_D in Figure 6.

The spatial dimension addresses overhead due to the actual cross-monitoring related computations and packet duplication. In the horizontal dimension, where one node monitors the forwarding of a packet of its neighbor in the principal communication path, it induces overhead at the monitoring node, but not the forwarding node [18]. In the orthogonal dimension cross-monitoring implies data redundancy, i.e., packet redundancy. Since a node can only cross-monitor if it contains the frame of reference, i.e., the packet it is verifying against, the space complexity in terms of packets and overhead associated with comparing packet content increases linear with the number of monitoring nodes. However, unlike in wired networks, in wireless networks due to the nature of broadcast, one does not have to pay the cost of packet duplication with respect to transmission. Thus, channel capacity is not affected as it would be in point-to-point distribution of the packets. Only if the paths are disjoint would communication introduce overhead.

This allows the determination of overhead for cross-monitoring in a join graph. First, the packet complexity of the cross-monitoring scheme is identical to that of the principal communication path only. Thus, there is no extra strain on the bandwidth. Second, the computational and storage complexities are linear in the number of participating monitors. Given the maximal clique C_i with vertex set V_i of the general join graph, then the monitoring overhead is $O(|V_i|)$. However, note that this is distributed over $|V_i| - 1$ nodes and the cardinality of V_i will probably be rather small, e.g., $|V_i| = [2, 4]$.

F. Fault-tolerance

Given a general join graph, one can determine the fault-tolerance of the communication between the source and destination. Tolerating a fault requires, in general, recover [6]. In the context of our model there are several 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. For example, in the horizontal technique used in [18], the omission essentially becomes a benign fault. However, malicious forwarding to a non-existing bogus node in order to avoid detection of the omission has the potential for strictly omissive asymmetric faults in networks with redundancy schemes.

Second, cross-monitoring based on comparison of packets in participating nodes constitutes spacial redundancy. As such, it is burdened with the 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.

Note that this is more powerful than the TCP model that relies on benign faults. Here, depending on the number of participants in a monitoring scheme, any fault type in Figure 1 can be detected.

G. Reliability Analysis

The reliability, $R(t)$, of a system is the probability that the system performs up to specifications during the entire time-interval $[0, t]$ [6]. In order to determine the reliability of a communication implemented as a join graph we will use the concept of Reliability Block Diagrams [6]. Specifically, the graph is a *series* graph, where each component is in turn a *parallel* construct, e.g., 1-of-N in case of benign faults or k-of-N for malicious faults, where k depends on the exact fault type assumed. It should be noted that this definition of reliability is standard in the dependability community, but it is arguably weak for modeling malicious human act, since it assumes a constant fail-rate, i.e., the reliability of a node is computed as $R(t) = e^{-\lambda t}$, where λ is the fail rate [6]. We want to stress that we consider this only a starting point and are currently working on reducing the dependance on a constant fail-rate using survival analysis. Only very few efforts have so far succeeded in eliminating the impact of λ, e.g. [15], and the application domain is still very limited.

In reference to Figure 5, 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 and the system is assumed homogeneous, the reliability $R_i(t)$ of the construct representing C_i, consisting of N_i nodes, is determined by

$$R_i(t) = 1 - \prod_{1}^{Ni}(1 - R(t)),$$

where $R(t)$ is the reliability of a single node. The reliability of the entire communication path from v_S to v_D is then

$$R_{SD}(t) = \prod_{i=S}^{D}(1 - \prod_{1}^{Ni}(1 - R(t))).$$

The terms of the inner product are defined for specific fault-models. Whereas here we assume a 1-of-N configuration, this can be changed, depending on the fault assumption, for each C_i, e.g., k-of-N. In the end, the formula above is only a series of parallel or k-of-N constructs.

To demonstrate the effect of cross-monitoring on the reliability, Figure 7 graphs the unreliability of three scenarios related to Figure 6 assuming $\lambda = 10^{-3}$ and assuming benign faults. First, only the principal communication path is considered. Next, cross-monitoring as shown in Figure 6 is assumed. Finally, the join graph is adapted to include cross-monitoring for v_7, thereby eliminating v_7 as a single point of failure. The results show that unreliability is greatly reduced when cross-monitoring is introduced. Figure 8 shows the unreliabilities approaching the mean time to failure, i.e., MTTF$= 1/\lambda$. As expected, as time moves toward the MTTF, the gains of the redundancy scheme levels off. The reason is that the example considers a system without replacement of failed nodes. In a real system there would be an attempt to replace failing nodes, e.g., redirecting troops or increasing node density in areas which experience higher than expected losses.

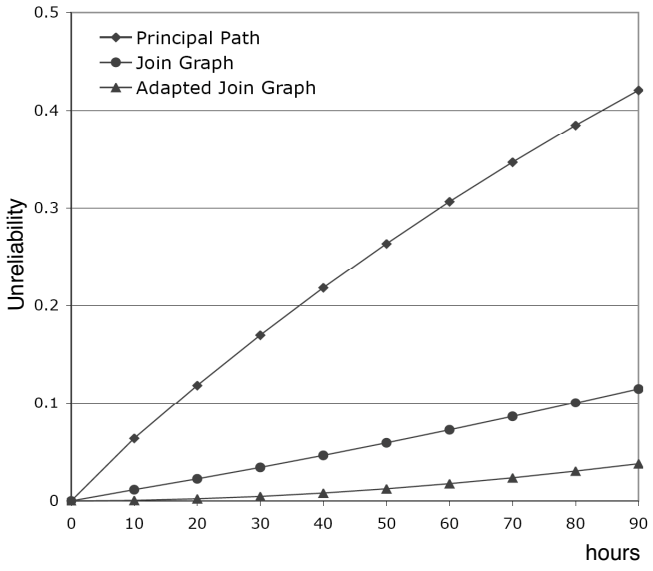

Fig. 7. Unreliabilities of Figure 6 - short term

III. Conclusion

To tolerate different fault types in wireless networks we have introduced a network model that is based on general join graphs, allowing for effective cross-monitoring. The orthogonal cross-monitoring overcomes the limitations of undetected faults in [3], [4], [18], [22] and can be the basis for fault recovery. The model intentionally does not focus on specific mechanisms that enhance security or reliability, as their existence is assumed, and considers faults that may occur despite all efforts. As such, it does not consider the cause of the fault but only its effect. This could be of special interest in critical environments where no assumptions can be made about malicious behavior or perhaps insider attacks. The assumption is taken that any protective mechanism could be compromised. This has important implications in sensitive environments where wireless nodes can fall in the hands of an enemy.

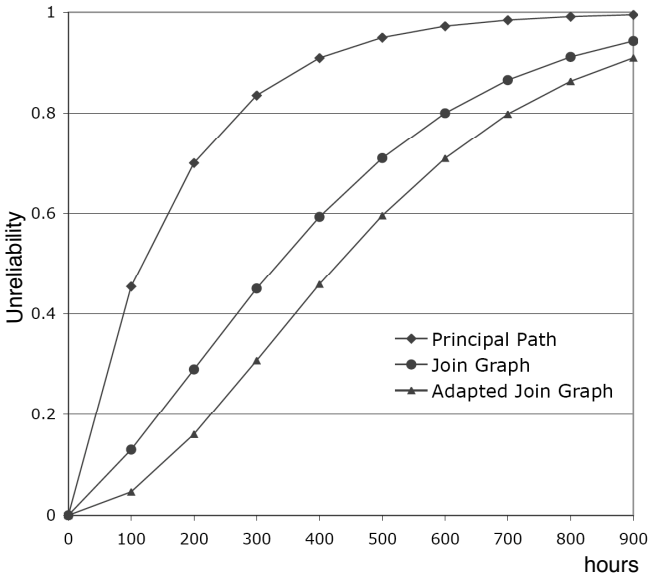

Fig. 8. Unreliabilities of Figure 6 - long term

REFERENCES

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

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

[3] S. Buchegger,C. Tissieres, and J.Y. Le Boudec, *A test-bed for misbehavior detection in mobile ad-hoc networks - how much can watchdogs really do?*, Sixth IEEE Workshop on Mobile Computing Systems and Applications, WMCSA 2004, pp. 102-111, 2-3 Dec. 2004.

[4] C. Chigan, and R. Bandaru, *Secure node misbehaviors in mobile ad hoc networks*, IEEE 60th Vehicular Technology Conference, VTC2004, Vol. 7, pp. 4730-4734, 26-29 Sept. 2004.

[5] D. Ganesan, R. Govindan, S. Shenker and D. Estrin, *Highly-resilient, energy-efficient multipath routing in wireless sensor networks*, Mobile Computing and Communications Review, Vol. 4, No. 5, October 2001.

[6] W. B. Johnson, *Design and Analysis of Fault-Tolerant Digital Systems*, Addison-Wesley Publishing Company, New York, 1989.

[7] R. M. Kieckhafer, C. J. Walter, A. M. Finn, and P. M. Thambidurai, *The MAFT architecture for distributed fault tolerance*, IEEE Transactions on Computers, 37(4), pp. 398-405, April 1988.

[8] A. Krings, *Agent Survivability: An Application for Strong and Weak Chain Constrained Scheduling*, Proc. 37^{th} Hawaii International Conference on System Sciences, (HICSS-37), paper STSSM02, pp. 1-8, January, 2004.

[9] A. Krings, *Primary-backup Link Scheduling for Wireless Networks Operating in Hostile Environments*, Scheduling Algorithms for new Emerging Applications, May 29th - June 2nd, CIRM, Marseille, France, 2006.

[10] L. Lamport, M. Pease, R. Shostak, *The Byzantine Generals Problem*, ACM Transactions on Programming Languages and Systems, Vol. 4, No. 3, pp. 382-401, July 1982.

[11] S. Lee and M. Gerla, *Split Multipath Routing with Maximally Disjoint Paths in Ad Hoc Networks*, Proc. of the IEEE ICC, pp. 3201-3205, 2001.

[12] C. Lee, X.-H. Lin, and Y.-K. Kwok, *A multipath ad hoc routing approach to combat wireless link insecurity*, Proceedings of IEEE International Conference on Communications (ICC), Vol. 1, pp. 448–452, April 2003.

[13] R. Leung, J. Liu, E. Poon, A. Chan and B. Li, MP-DSR: A QoS-aware Multi-path Dynamic Source Routing Protocol for Wireless Ad-Hoc Networks, *Proc. 26th Annual IEEE Conference on Local Computer Networks*, LCN 2001, pp. 132-141, 2001.

[14] Deying Li, Xiaohua Jia, and Hongwei Du, *QoS Topology Control for Nonhomogenous Ad Hoc Wireless Networks*, Volume 2006, Article ID 82417, 10 pages, 2006.

[15] Y. Liu and K. S. Trivedi, *A General Framework for Network Survivability Quantification*, Proceedings of the 12th GI/ITG Conference on Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB), Dresden, Germany, Sep. 2004.

[16] H. Liu and D. Raychaudhuri, *Label Switched Multi-path Forwarding in Wireless Ad-Hoc Networks*, Proceedings of the 3rd Intl Conf. on Pervasive Computing and Communications Workshops, (PerCom 2005 Workshops), pp. 248-252, 2005.

[17] M. K. Marina and S. R Das, *On-Demand Multipath Distance Vector Routing for Ad Hoc Networks*, Proc. of the International Conference for Network Protocols (ICNP), Riverside, USA, pp. 14-23, 2001.

[18] Sergio Marti, T. J. Giuli, Kevin Lai and Mary Baker, *Mitigating routing misbehavior in mobile ad hoc networks*, Mobile Computing and Networking, pp. 255-265, 2000.

[19] F. J. Meyer, and D. K. Pradhan, *Consensus with Dual Failure Modes*, IEEE Transactions on Parallel and Distributed Systems, Vol. 2, No. 2, pp. 214-222, April, 1991.

[20] A. Mishra, K. Nadkarni, and A. Patcha, *Intrusion detection in wireless ad hoc networks*, IEEE Wireless Communications, Vol. 11, Issue 1, pp. 48-60, Feb 2004.

[21] S. Mueller, R. P. Tsang, and D. Ghosal, *Multipath Routing in Mobile Ad Hoc Networks: Issues and Challenges*, MASCOTS 2003, LNCS 2965, Springer-Verlag, pp. 209234, 2004.

[22] A. Patcha, and A. Mishra, *Collaborative security architecture for black hole attack prevention in mobile ad hoc networks*, Proc. Radio and Wireless Conference, RAWCON '03, pp. 75-78, 10-13 Aug. 2003.

[23] F.B. Schneider, *Byzantine Generals in Action: Implementing Fail-Stop Processors*, Computer Systems, Vol. 2, No. 2, pp. 145-154, 1984.

[24] F.B. Schneider, *Towards Fault-tolerant and Secure Agentry*, Proc. of the 11th International Workshop on Distributed Algorithms, Saarbrucken, Germany. September, 1997.

[25] J.R. Sklaroff, *Redundancy Management Technique for Space Shuttle Computers*, IBM Journal on Research and Development, Vol. 20, No. 1, pp. 20-28, Jan. 1976.

[26] N. Song, L. Qian, and X. Li, *Wormhole attacks detection in wireless ad hoc networks: a statistical analysis approach*, Proc. 19th IEEE International Parallel and Distributed Processing Symposium, 8 pages, 4-8 April, 2000.

[27] J.P.G. Sterbenz, R. Krishnan, R.R. Hain, A.W. Jackson, D. Levin, R. Ramanathan, and J. Zao, *Survivable mobile wireless networks: issues, challenges, and research directions*, ACM Workshop on Wireless Security (WiSe), Atlanta, GA, USA, 1:31–40, 28 September 2002.

[28] F. Stajano and R. Anderson, *The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks*, Security Protocols, 7th International Workshop Proceedings, Lecture Notes in Computer Science, Springer-Verlag, pp. 172-194, 1999.

[29] J. Tang , G. Xue and W. Zhang, *Reliable Routing in Mobile Ad Hoc Networks Based on Mobility Prediction*, Proc. First IEEE Intl. Conference on Mobile Ad Hoc and Sensor Systems, MASS2004, Fort Lauderdale, Florida, pp. 466-474, Oct. 24-27, 2004.

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

[31] N. Thanthry, et.al., *TCP-M: Multiflow Transmission Control Protocol for Ad Hoc Networks*, EURASIP Journal on Wireless Communications and Networking, Article ID 95149, 16 pages, 2006.

[32] Jay J. Wylie, et.al., *Selecting the Right Data Distribution Scheme for a Survivable Storage System*, Technical Report, CMU-CS-01-120, Carnegie Mellon University, May 2001.

[33] Z, Ye, S. V. Krishnamurthy, and S. K. Tripathi, *A routing framework for providing robustness to node failures in mobile ad hoc networks*, Ad Hoc Networks 2, Elsevier publishing, pp. 87-107, 2004.