CHAPTER 7

Case Study, Continued: Levels 1 to 3

7.1 Level 1: The First Refinement Level

I select the HazHapp Lost(msg) as the driver of the refinement at Level 1. Structure is added to messages in order to say what it means for a msg to be Lost, and maybe to perform a causal system analysis of lost messages; to identify all of the causal factors and their interrelations which lead to or may lead to a lost message.

7.1.1 Moving to Level 1: Structuring Messages and Message-Passing

We could assume a bus protocol which does not rely on NICs synchronising clocks, but is opportunistic, event-triggered, as in Ethernet. An opportunistic bus protocol works roughly as follows. Say, NIC1 wants to transmit msg1. NIC1 first tests the Bus to see if there is a message already on it. If not, NIC1 may begin to transmit. It may be that another NIC, say NIC2, has already started transmitting msg2, but that msg2's wave front has not yet reached the Bus connector of NIC1, and so was not sensed as NIC1 tested the Bus. NIC1 starts transmitting msg1, and the wave front of msg1 and msg2 will subsequently collide somewhere. We may take the result of two colliding messages always to be nonsense. Such an opportunistic bus protocol must have a method of collision detection, collision communication (certainly to the senders and receivers of the colliding messages, maybe to every NIC), and resolution. In Ethernet, resolution involves NIC1 and NIC2 retrying to transmit, after waiting distinct or likely-distinct periods of time.

An alternative transmission protocol, called a time-triggered or round-robin protocol, has NICs transmitting in specified time slots reserved for that NIC. It requires the NICs to synchronise their internal clocks, so that they all know what bus-time it is. They can do this within certain error bounds using well-analysed algorithms. Time slots are defined by the protocol, one slot for each possible transmission. So, if we had 40 NICs, we could define a cycle size of 40 time slots, with NIC-k putting its message, if it has one, on the bus in the k'th time slot of each cycle.

Round-robin scheduling has no collisions, by construction, so no need for collision detection and resolution, provided that the clock synchonisation is reliable (the boundaries of adjacent time slots may be separated from each other by a period equal to or greater than the error bound). However, there may be NICs that need to transmit only rarely, and other NICs which transmit in every cycle. Most of the slots of the rare transmitters will not be used in the scheme of the last paragraph. Thus there can arise a problem of efficiency: a NIC which needs to transmit must wait for its slot, and in the meantime a lot of free capacity flows by as it is waiting. Furthermore, there may be problems with recognising or rejecting messages which near timeslot boundaries [1].

This is not the place to discuss the merits and demerits of various bus protocols, but the information thus far will help us specify the vocabulary of Level 1.

7.1.2 Level 1 General Definitions and Meaning Postulates

Packets

Time-triggered or event-triggered messages are often decomposed into packets. Packets are "individually wrapped" portions of a message of more-or-less uniform size (the transmission-protocol information added to the content of a message is called a "wrapper"). Handling similar-size packets has advantages for the efficiency of a transmission protocol. A packet will be a new object, and we shall need a new relation saying that a packet constitutes part of a message:

$$\text{ConstitutesPartOf(pkt,msg)}$$

A packet may be an arbitrary division of a message, such as the pages of a book are an arbitrary division of its paragraphs, but we could also insist that a packet is a meaningful part of a message; that is, that it contains specific fields, and all the information in each field, of the message. If the fields are expressed as <attribute,value> pairs, such a packet would contain whole <attribute,value> pairs; none of them is split across more than one packet. We can introduce a predicate to say that a specific field, which we may take abstractly to be an <attribute,value> pair, is contained in a specific packet:

$$\text{FieldContainedIn(field,pkt)}$$

We may also be specific about a field with the meaning postulate and constraint[1]:

$$\text{field}(x) \Leftrightarrow \exists\ \text{a:attribute}\ \exists\ \text{v:value.}\ x = \langle a,v\rangle$$
$$\text{field}(x)\ \text{AND}\ x = \langle a,v\rangle \Rightarrow \text{AttributePart}(x) = a\ \text{AND ValuePart}(x) = b$$

We can also speak of a field being contained in a msg, by means of the meaning postulate

$$\text{FieldContainedIn}_M\text{(field,msg)}$$
$$\Leftrightarrow$$
$$\exists\ \text{p:pkt (FieldContainedIn(field,p) AND ConstitutesPartOf(p,msg))}$$

We can also define Sending(pkt,NIC) and Sent(pkg,NIC) analogously to the terms for msg. I leave that for the exercises.

Checksums

Where information is required to be reliable, it is usual to include some reliability checks, for example, a parity check, or some more complex CRC calculation. We refer to such a calculation generically as a Checksum. I introduce the new property of a pkt or a msg, Valid, to say that the Checksum of the pkt or msg is valid.

More Detailed NIC Behaviour

We can say something more now, informally, about what a NIC must do. At Level 0, a NIC was unanalysed, an "atomic object". At Level 1 we can now say that, in its Sender

1 Note that here attribute and value are object types, whereas AttributePart and ValuePart are functions of a field which give the attribute, respectively value, parts of the field.

Object Types	Properties
pkt	Valid
field	AttributePart ValuePart
attribute	primitive
value	primitive
Checksum	primitive
msg	Valid

Figure 7.1: Level 1 Object Types and Properties

Relations	Object Types
ConstitutesPartOf	pkt,msg
FieldContainedIn	field,pkt
FieldContainedIn$_M$	field,msg
Sending	pkt,NIC
Sent	pkt,NIC

Figure 7.2: Level 1 Relations

role, it decomposes a msg into pkts to put on the Bus. It puts the <attribute,value> pairs constituting a specific msg into pkts of more or less uniform size, and then puts those pkts on the Bus in some order. For all we know, it composes <attribute,value> from raw sensor data.

In its Receiver role, a NIC performs the task of Receiving all the pkts comprising a particular msg and sending all the <attribute,value> pairs comprising that msg onwards to their intended destinations (not specified in the Communications Bus system). We could pack all this information about the actual tasks of a NIC into Level 1. However, in line with experience in refinement, we choose to perform one refinement task at once, and leave other refinement tasks to a subsequent refinement step.

Certain protocols, such as the Internet protocol suite TCP/IP, take different design decisions from those I am taking here. For example, TCP/IP divides data into packets without paying attention to where the packet boundary occurs with respect to meaningful content (fields). TCP/IP chops data up, as the pages of a book chop up the paragraphs of the author. TCP relays the packet with a sequence number, equivalent to the page of a book, which is used to reassemble the entire data stream from its packets at the destination. Representating fields, as here, as <attribute,value> pairs, along with the requirement that such pairs are contained within precisely one packet and not split, obviates the need for any sequence numbering. The meaningful contents of a msg are just the collection of the meaningful contents of its pkts; order is not relevant.

We can now derive another meaning postulate, for Received. A msg has been Received by a NIC just in case all its <attribute,value> pairs have been read by the NIC. This is the case precisely when all pkts constituting the msg have been read by the NIC and the NIC has recomposed all pkts into the msg of which they constitute part. This MP uses the new primitive term Composed.

$$\begin{gathered}\text{Received(msg,IntendedReceiver(msg))}\\ \Leftrightarrow\\ \forall\ \text{p:pkt (ConstitutesPartOf(p,msg)} \Rightarrow\\ \text{Received(p,IntendedReceiver(msg)))}\\ \text{AND Composed(msg,IntendedReceiver(msg)))}\end{gathered}$$

$$\boxed{\text{field(x) AND x} = \text{<a,v>} \Rightarrow \text{AttributePart(x)} = \text{a AND ValuePart(x)} = \text{b}}$$

Figure 7.3: Level 1 Partial Meaning Postulate

7.1.3 Level 1 Hazard Analysis for Lost(Msg)

We can now introduce a meaning postulate for the HazHapp Lost(msg). First I introduce a shorthand, overloading the meaning of Sent:

$$\begin{gathered}\text{Sent(msg)} \Leftrightarrow \exists\ \text{NIC.Sent(msg,NIC)}\\ \text{Sent(pkt)} \Leftrightarrow \exists\ \text{NIC.Sent(pkt,NIC)}\end{gathered}$$

I have included these in the initial list of meaning postulates in Figure 7.4. Using this shorthand, recall, as in Figure 7.5, that Lost(msg) is equivalent to

Term	Definition
field(x)	$\exists$ a:attribute $\exists$ v:value. x = <a,v>
FieldContainedIn$_M$(field,msg)	$\exists$ p:pkt. (FieldContainedIn(field,p) AND ConstitutesPartOf(p,msg))

Received(msg,IntendedReceiver(msg))	$\forall$ p:pkt (ConstitutesPartOf(p,msg) $\Rightarrow$
	Received(p,IntendedReceiver(msg)))
	AND
	Composed(msg,IntendedReceiver(msg))

Sent(msg)	$\exists$ NIC.Sent(msg,NIC)
Sent(pkt)	$\exists$ NIC.Sent(pkt,NIC)

Figure 7.4: Some Level 1 Meaning Postulates

Sent(msg)
AND NOT (Received(msg,IntendedReceiver(msg)))
AND NOT (On(msg,Bus))

Term	Definition
Sent(msg,NIC)	NOT Sending(msg,NIC) AND $\Diamond_P$(Sending(msg,NIC))
Sender(msg1) = NIC1	Sent(msg1,NIC1)
Received(msg,NIC)	$\Diamond_P$(Receiving(msg,NIC)) AND NOT On(msg,Bus)
Lost(msg)	Sent(msg) AND NOT(Received(msg,IntendedReceiver(msg))) AND NOT(On(msg,Bus))
OriginalContent(msg1) = Y	$\Diamond_P$(Sending(msg1,NIC1) AND Content(msg1) = Y)
Corrupted(msg1)	NOT(OriginalContent(msg1) = Content(msg1))

Figure 7.5: Relevant Meaning Postulates from Level 0

This constrains in useful ways what constitutes losing a msg, as follows. Let us consider first errant pkts. How could a pkt not successfully reach its destination? Consider its temporal progress; it will get pulled up short somewhere. First, suppose the pkt is not put on the Bus by its NIC – it was supposed to be put on the Bus, but it wasn't. We have some vocabulary to say this already, but not yet to say to which NIC a pkt belongs. We extend the meaning of Sender and IntendedReceiver to pkts[1]

$$\text{Sender(pkt)} = \text{Sender(msg) WHERE ConstitutesPartOf(pkt,msg)}$$
$$\text{IntendedReceiver(pkt)} = \text{IntendedReceiver(msg) WHERE ConstitutesPartOf(pkt,msg)}$$

One possibility for non-reception of a msg is that not all the pkts which Constitutes-PartOf the msg were put on the Bus by their Sender NIC. But this is not the case with a Lost(msg) – the meaning postulate for Lost contains a conjunct which says the msg was Sent. A message was Sent if all its pkts were Sent, as per the meaning postulate:

$$\text{Sent(msg)} \Leftrightarrow \forall \text{ p:pkt (ConstitutesPartOf(p,msg)} \Rightarrow \text{Sent(p))}$$

If a pkt doesn't make it onto the Bus from its NIC, this does not constitute part of its associated msg being Lost, because the msg was thereby not Sent. We do not want to just ignore this failure mode, though. In fact, we have a HazHapp which already covers it. Putting pkts correctly on the bus concerns the HW or SW Integrity of the NIC. NOT Integrity(NIC) is already amongst the HazHapps of Level 0, covering this case also.

Term	Definition
Sender(pkt)	Sender(msg) WHERE ConstitutesPartOf(pkt,msg)
IntendedReceiver(pkt)	IntendedReceiver(msg) WHERE ConstitutesPartOf(pkt,msg)
Sent(msg)	$\forall$ p:pkt (ConstitutesPartOf(p,msg) $\Rightarrow$ Sent(p))

Figure 7.6: More Level 1 Meaning Postulates

The definition of Lost implies all of the pkts of the msg were put on the Bus. It is not there now (the last conjunct of Lost), and the msg wasn't Received by its

1 We can do this, since the Object Types are logically sorts. We can have a property range over two sorts and not just one. I omit details because they are not really relevant to the subject matter.

IntendedReceiver. If it is not On the Bus now, either the time has gone by during which the waves constituting the pkts will have taken to reach IntendedReceiver, or we are still within the time within which some pkt should still be in transit, but somehow it is not: a Lost(pkt).

That can only happen in three ways.

1. If the bus has been physically compromised, broken or at least damaged enough to attentuate the signal so that it is indistinguishable from background, then a pkt or more than one will disappear in this sense
2. If there is a collision, and this collision goes undetected, then the pkt as information will have disappeared; its contents will have been incorporated into the noise resulting from the collision.
3. The pkt has a Checksum which is not VALID when checked by the IntendedReceiver: NOT Valid(pkt). This has two sub cases:
   - The Checksum is really not valid
   - The Checksum is valid but the IntendedReceiver made a mistake. In this case, NOT Integrity(IntendedReceiver).

Event type (1) is encapsulated in the HazHapp NOT Integrity(Bus). Event type (2) is a possibility not yet covered. Let us denote it by the new primitive

UndetectedCollision(pkt).

From this discussion we may conclude

$$\begin{gathered}\text{Lost(pkt)}\\ \Rightarrow\\ \text{NOT Integrity(Bus) OR UndetectedCollision(pkt)}\\ \text{OR NOT Valid(pkt) OR NOT Integrity(IntendedReceiver)}\end{gathered}$$

The final case to consider is the one in which the all pkts reach their destination, but the msg has not been formally received:

$$\begin{aligned}&\forall\ \text{p:pkt.(}\quad \text{ConstitutesPartOf(p,msg)}\\ &\qquad\qquad \text{AND IntendedReceiver(p) = IntendedReceiver(msg)}\\ &\qquad\qquad \text{AND Received(p,IntendedReceiver(p)))}\\ &\text{AND NOT Received(msg,IntendedReceiver(msg))}\end{aligned}$$

How could this happen? It could be that NOT Valid(Checksum(msg)) even though all the pkt Checksums were Valid. This could happen, again, because the Checksum was really not Valid, or because the NIC performed a faulty Checksum computation, again NOT Integrity(IntendedReceiver(msg)). Or it could be that the message was not recomposed appropriately by the NIC, NOT Integrity(IntendedReceiver(msg)). It could in fact be that one of the pkts contained a field which was modified, and the Checksum of the pkt was modified along with it, to cohere, so that the pkt checked out on receipt. But then the msg Checksum wouldn't check out - a case we just considered.

We may want to separate these cases - indeed we might want to choose methods to implement Checksum in a later refinement step to make the chances of a field being modified along with its Checksum to be vanishingly small. If not, we would have to consider

$$\begin{gathered}\text{Modified(pkt)}\\ \Leftrightarrow\\ \text{Content(pkt)} \neq \text{OriginalContent(pkt) AND Valid(Checksum(pkt))}\end{gathered}$$

but I do not do so here. Making Checksums do their job seems like the right way to go. We thus have the following partial meaning postulate for Lost(msg):

$$\begin{gathered}\text{Lost(msg)} \Rightarrow\\ \exists\ \text{p:pkt ConstitutesPartOf(p,msg) AND Lost(p)}\\ \text{OR NOT Valid(Checksum(msg))}\\ \text{OR NOT Integrity(IntendedReceiver(msg))}\end{gathered}$$

The partial meaning postulates for the msg and pkt versions of Lost are summarised in Figure 7.7.

7.1.4 Level 1 HazHapp Avoidance and Mitigation

Procedures with which to deal with an occurrence of Lost(pkt) in a msg, that is, a message which contains lost or corrupted packets, are ubiquitous in network protocol engineering. One of the most well-known, also used in the Internet reliable-transmission protocol TCP, is called the Sliding Windows protocol. All relevant features of Sliding Windows are known, including reliability rates for various versions. Here is not the place to discuss it; there are many texts, for example [2].

Lost(pkt) $\Rightarrow$	NOT Integrity(Bus) OR UndetectedCollision(pkt) OR NOT Valid(pkt) OR NOT Integrity(IntendedReceiver)
Lost(msg) $\Rightarrow$	$\exists$ p:pkt ConstitutesPartOf(p,msg) AND Lost(p) OR NOT Valid(Checksum(msg)) OR NOT Integrity(IntendedReceiver(msg))

Figure 7.7: Level 1 Partial Meaning Postulates for Lost

Procedures to deal with Checksums sufficient to identify Modified packets may be found, along with reliability estimates, in textbooks on error-detecting and error-correcting codes.

7.1.5 Summary of Level 1 Results

A key feature of CFA is the careful control of the expression of system characteristics, including hazards and failures, by using a controlled vocabulary along with features of logical languages. A second key feature is the use of refinement, to provide more detail about the system and simulatneously extend the vocabulary, guiding this refinement by the classification of hazards, and steps to identify and catalog mitigation and avoidance.

We started by considering a generic communications bus for a road vehicle which uses this bus for control and sensorics, and the hazards that could arise through use of this bus for these purposes. We identified at Level 0 necessary vocabulary with which to talk about these hazards, and we identified hazards, but could not analyse them, since they appeared at Level 0 as primitive vocabulary.

In CFA, it is recommended not to bite off too much at once: not to attempt to introduce in one step new concepts and vocabulary to handle all HazHapps identified at Level 0. We thus proceed to Level 1 by picking one HazHapp and analysing it further.

At Level 1, we considered in detail the HazHapp Lost(msg). We were able to classify some of the phenomena which might lead to a lost message under other HazHapps, namely NOT Integrity(NIC) and NOT Integrity(Bus). One part of the intuitive phenomenon associated with Lost(msg) that was not assimilated to these other HazHapps turned out to have much in common with the phenomena associated with another HazHapp Corrupted(msg), and so we extended consideration to the similar characteristics involved also in Corrupted(msg). The remaining part of the phenomenon of Corrupted(msg) was identified through introducing a new property.

The vocabulary was realigned to relinquish use of Lost and Corrupted in favor of the use of VisiblyCorrupted and Modified, which predicates align exactly with known mitigation and avoidance procedures with well-known reliability characteristics, of which use can be made in the risk analysis which will follow the hazard analysis.

At this Level, we have identified the following HazHapps which have not been handled at Level 1 and therefore will become the subject of further refinement steps:

Unanalysed HazHapps for Further Refinement After Level 1
NOT Integrity(Bus)
NOT Integrity(NIC)
InappropriateReceiver(msg,NIC)
OutsideInterval(msg)
PartlyOutsideInterval(msg)
IntermittentlyAttached(NIC)
CorruptedSending(msg,NIC)
CorruptedReceiving(msg,NIC)
UndetectedCollision(pkt)

Figure 7.8: Identified Levels 0 and 1 HazHapps to be Analyzed Later

Refinement continues. To identify the next Level (Level 2), we consider the HazHapps in Figure 7.8, and use these as guidance.

Term	Definition
field(x)	∃ a:attribute ∃ v:value. x = <a,v>
FieldContainedIn$_M$(field,msg)	∃ p:pkt. (FieldContainedIn(field,p) AND ConstitutesPartOf(p,msg))

Received(msg,IntendedReceiver(msg))	∀ p:pkt (ConstitutesPartOf(p,msg) ⇒
	Received(p,IntendedReceiver(msg)))
	AND
	Composed(msg,IntendedReceiver(msg))

Sent(msg)	∃ NIC.Sent(msg,NIC)
Sent(pkt)	∃ NIC.Sent(pkt,NIC)

Sender(pkt)	Sender(msg)
	WHERE ConstitutesPartOf(pkt,msg)

IntendedReceiver(pkt)	IntendedReceiver(msg)
	WHERE ConstitutesPartOf(pkt,msg)

Sent(msg)	∀ p:pkt (ConstitutesPartOf(p,msg) ⇒ Sent(p))

Figure 7.9: All Level 1 Meaning Postulates

field(x) AND x = <a,v>	⇒	AttributePart(x) = a AND ValuePart(x) = b
Lost(pkt)	⇒	NOT Integrity(Bus) OR UndetectedCollision(pkt) OR NOT Valid(pkt) OR NOT Integrity(IntendedReceiver)
Lost(msg)	⇒	∃ p:pkt ConstitutesPartOf(p,msg) AND Lost(p) OR NOT Valid(Checksum(msg)) OR NOT Integrity(IntendedReceiver(msg))

Figure 7.10: All Level 1 Partial Meaning Postulates

A pkt consists of an integral number of fields

Figure 7.11: New Level 1 Assumptions

7.2 Level 2 Refinement

There is one HazHapp in Figure 7.8 identified with the Bus, namely NOT Integrity(Bus). We have already considered the possibility of UndetectedCollision, and there is an exercise to identify a reliability mitigation for those. Other pkt losses associated with NOT Integrity(Bus) for which reliability countermeasures remain to be formulated are those associated with physical-electrical failures such as short-circuits and breaks in the cabling. These can also be handled through known techniques from electrical engineering. A loss of Integrity of the Bus cannot be ruled out completely, because physical activities such as someone deliberately slicing a cable cannot be ruled out, but it can be detected, for example by the countermeasure in Figure 7.13.

New Objects	New Properties	New Relations
None	None	None

Figure 7.12: New Entities Added for Level 2

HazHapp	Reliability Countermeasure
NOT Integrity(Bus)	Electrical-anomaly detection, e.g., arc-fault circuit interrupters

Figure 7.13: Partial List of Level 2 HazHapps and RelReqs

Unanalysed HazHapps for Further Refinement and Countermeasure Formulation After Level 2
NOT Integrity(NIC)
InappropriateReceiver(msg,NIC)
OutsideInterval(msg)
PartlyOutsideInterval(msg)
IntermittentlyAttached(NIC)
CorruptedSending(msg,NIC)
CorruptedReceiving(msg,NIC)

Figure 7.14: Partial List of Level 2 HazHapps for Later Analysis

7.3 Deciding on Level 3

There are only two sorts of objects, msg and NIC, remaining in the cumulative list of HazHapps to be analysed and mitigated. The HazHapps OutsideInterval and PartlyOutsideInterval refer to deadlines and timing of messages. The other HazHapps refer either to a NIC alone, or to processing of a msg through a NIC. Since a msg is a data object, and not an active agent (it does not execute actions), we can usefully regard CorruptedSending and CorruptedReceiving as hazards associated with unsuccessful actions of the NIC, and therefore reflecting on the Integrity of the NIC, which in general terms means the ability of the NIC to carry out its required functions in an appropriate time. There are two ways Integrity might therefore fail. One way is that a required processing function is not carried out. The other way is that appropriate timing is not achieved. That appropriate timing constraints are not fulfilled also reflects on msg deadlines, which in turn are associated so far with OutsideInterval(msg) and PartlyOutsideInterval(msg). In the other direction, that a msg does not encounter its receiving NIC in time does not necessarily reflect on the Integrity of the receiving NIC. So there appears to be an asymmetry, in that considering the Integrity of a NIC may well lead to modification of the HazHapps designated as OutsideInterval and PartlyOutsideInterval, but considering these latter will not reflect in any particular way on the Integrity of the receiving NIC.

These reflections point us firmly in the direction of considering Integrity(NIC), and refining the required operations of the NIC at the next refinement level, Level 3.

Considering msg timing constraints will occur, then, at a further level of refinement beyond this.

The exact functioning of the NICs has at this point not been addressed at all. Nothing has yet been said about what a NIC shall do. Indeed, nothing has yet been said about the protocol under which the Bus will run and which parts of the protocol will be the responsibility of the NICs to ensure. If the Bus runs under a purely event-driven protocol, such as Ethernet, timing constraints on message reception are driven purely by the timing constraints of the attached equipment, whereas if the protocol is partly or completely time-triggered, there are possibilities for a message to arrive out-of-slot or partly-out-of-slot, thereby manifesting a HazHapp. These phenomena are purely internal to the communications under time-triggered protocols and do not concern physical constraints on the attached equipment.

We quit here, leaving the HazAn at Level 3 and beyond to the reader.

7.4 Conclusion

We have performed part of a Causal Fault Analysis of a generic communication bus which was considered to have attached sensors and actuators through management devices called NICs, say for a road transport vehicle. CFA proceeds methodologically similarly to OHA, but is concerned with identifying faults and failures rather than hazards. Any functional failure could lead to a hazard in some application, depending upon the application, and I called the functional failures HazHapps to emphasise this connection. Failure analysis, like hazard analysis, is still a process in which, to put it bluntly, one is asked to sit down and think of as many failures, respectively hazards, that can befall a system in operation as possible. The goal is to think of and write down as many as one can. OHA and CFA use formal or semi-formal refinement and control of vocabulary using OPRA.

We used one application of HAZOP, in order to start the Failure Analysis at Level 0. The HAZOP principle of groupthink was not applied; although checking through colleagues is necessary as always, we have found that the major benefits of OHA and CFA come from the control that semi-formal refinement brings, and not primarily through subtleties in application of HAZOP guidewords.

7.5 Exercises

1. We considered formulating Sending and Sent for both msgs and pits. Formulate a single OPRA definition of each which accommodates both msgs and pkts.
2. We considered formulating Lost for both msgs and pkts. Formulate a single OPRA definition of each which accommodates both msgs and pkts.
3. Formulate the partial meaning postulates for Lost as one meaning postulate which applies to both msgs and pkts. Do you find it easier or harder to read than the separately formulated ones in Figure 7.7?
4. We considered in passing that packets should have a more or less uniform size. This comes from experience with computer networks. Since a message consists of an integral number of packets, and the packets have more or less uniform size, it seems reasonable to take the Size of a msg to consist of the number of packets which constitute it. Formulate an OPRA definition.
5. We considered briefly a property Modified of a pkt. Formulate a reliability requirement, a RelReq, which would rule this case out in all but vanishingly few cases.
6. What can we do about undetected collisions on the Bus? Formulate a reliability requirement, and determine what mitigation measures exist to fulfil it. You might have to consult a book such as [2].
7. What would you choose to analyse at refinement Level 3? Do so.

Bibliography

[1] Kevin Driscoll, Brendan Hall, Håkan Sivenkrona and Phil Zumsteg, *Byzantine Fault Tolerance, from Theory to Reality*, in Computer Safety, Reliability and Security, Proceedings of the 22nd International Conference, SAFECOMP 2003, Lecture Notes in Computer Science volume 2788, Springer-Verlag, 2003. Available from https://www.cs.indiana.edu/classes/p545-sjoh/post/lec/fault-tolerance/Driscoll-Hall-Sivencrona-Xumsteg-03.pdf , accessed 2017-06-14.

[2] Andrew S. Tanenbaum and David J. Wetherall, Computer Networks, 5th edition, Pearson, 2011.