Ubiquitous faults

T-79.4001 Seminar on Theoretical Computer Science

Tero Pietiläinen

4.4.2007

Outline

Outline

Ubiquitous faults

- Majority of failures have mostly transient and ubiquitous nature
- They are also called *dynamic faults* or *mobile faults*
- They are much more difficult to handle than localized faults

Outline

Communication

- In synchronous networks silences are expressive as observed in chapter 6
- Let us define *communication* as follows: Given an entity *x* and neighbour *y* in *G*, at each time unit *t*, a *communication* from *x* to *y* is a pair <α, β> where α denotes what is sent by *x* and β what is recieved by *y* from *x* at time *t* + 1.
- We denote by $\alpha = \phi$ that at time *t*, *x* didn't send a message to *y*. By $\beta = \phi$ we denote that at time *t* + 1, *y* didn't recieve any message from *x*.

Communication Faults

- A communication <α, β> is *faulty* if $\alpha \neq \beta$
- Three types of faulty communication:
  - Omission, ($\alpha \neq \phi = \beta$)
  - Addition, ($\alpha = \phi \neq \beta$)
  - Corruption, ($\phi \neq \alpha \neq \beta \neq \phi$)
- These three types of faults are quite incomparable with each other in terms of danger
- The presence of all three fault types creates what is called a Byzantine faulty behavior

Agreement Problem, Agree(p)

- The goal will be to determine if and how a certain level of agreement can be reached in spite of certain number F of dynamic faults of a given type τ occuring at each time unit.
- As the faults are dynamic, the set of faulty communications may change at each time unit.
- We are mainly interested in the following agreement problems:
  - *Unanimity*, $p = n$
  - *Strong majority*, $p = \lceil \frac{n}{2} \rceil + 1$
- Any Boolean agreement requiring *less* than strong majority can be trivially reached without any communication

Outline

Limits for Reaching Majority

- In a network $G = (V, E)$ with maximum node degree $\deg(G)$
  - 1. With $\deg(G)$ omissions per cycle, strong majority cannot be reached.
  - 2. If the failures are any mixture of corruptions and additions, the same bound holds
  - 3. In the Byzantine case strong majority cannot be reached with $\lceil \deg(G)/2 \rceil$ faults

About the proof

- The proof is obtained a bit similary as the Single-Fault disaster, but
  - We are now in *synchronous* enviroment
  - Delays are unitary; we cannot employ arbitrary long delays
  - Omissions are detectable
  - It follows that the proof is more complicated
- The *problem*
  - Each entity x has an *input register* I_x and a write once ouput I_o
  - Initially $I_x \in \{0,1\}$ and all output registers set to the same value b $\notin \{0,1\}$
  - Goal: at least $p > \lceil n/2 \rceil$ entities set their output registers, in finite time, to the same value d

Definitions (1/4)

- *Internal state* $s_i(C)$ of an entity: values of registers, global clock, program counters and internal storage
- *Configuration* C: Internal states of all entities at a given time. A *configuration* has *decision value* v if at least p entities are in v-decision state
- *Message array* $\Lambda(C)$: Composed of n^2 entries as follows
  - If x_i, x_j are neighbours then $\Lambda(C)[i,j]$ contains message sent by x_i to x_j
  - Else $\Lambda(C)[i,j] = *$, where $*$ is distinguished symbol

Definitions (2/4)

- *Transmission matrix* τ for $\Lambda(C)$: descripes the actual communication by means of another $n \times n$ array
  - If x_i, x_j are neighbours then $\tau[i,j] = (\alpha, \beta)$, where $\alpha = \Lambda(C)[i,j]$ and β is what x_j actually receieves
  - Else $\tau[i,j] = (*, *)$
  - Many transmission matrices are possible for the same Λ. Let $\mathcal{T}(\Lambda)$ denote the set of all possible τ for Λ
- Let $R^1(C) = R(C) = \{\tau\{C\} : \tau \in \mathcal{T}(\Lambda(C))\}$ be the set of all possible conigurations resulting from C in one step.
- Similary let $R^k(C)$ be the set of all possible conigurations resulting from C in k>0 steps.
- Let $R^*(C)$ be the set of configurations reachable from C

Definitions (3/4)

- A configuration is *v-valent* if there exists a $t \geq 0$ such that all $C' \in R^t(C)$ have decision value v
- A configuration is bivalent if there exists in $R^*(C)$ both a 0-valent and a 1-valent configuration
- If two configurations C' and C'' differ only in the internal state of entity x_j we say that the configurations are j-adjacent and we call them adjacent if they are j-adjacent for some j
- We call a set S of events *j-adjacency preserving* if for any two j-adjacent configurations C' and C'' there exists in S τ' and τ'' such that $\tau'(C')$ and $\tau''(C'')$ are j-adjacent.
- We call S *adjacency preserving* if it is j-adjacent for all j

Definitions (4/4)

- A set S of events is *continuous* if for any C and for any τ', $\tau'' \in S$ for $\Lambda(C)$ there exists a finite sequence $\tau_0, ..., \tau_m$ of events in S for $\lambda(C)$ such that $\tau_0 = \tau'$, $\tau_m = \tau''$, and $\tau_i(C)$ and $\tau_{i+1}(C)$ are adjacent, $0 \leq i < m$
- A set S of events is F-admissible if for each message matrix Λ, there is an event $\tau \in S$ for Λ that contains at most F faulty transmissions; furthermore there is an event in S that contains exactly F faulty transmissions

Few properties to help with the proof

- Properties that follow from the definitions
  - If an entity is in the same state in two different configurations, then it will send the same messages in both configurations
  - If an entity is in the same state in two different configurations and it receives the same messages in both configurations, then it will enter the same state in both resulting configurations

A Theorem to Help with the Proof (1/3)

- Let $\mathcal{P}(P, S)$ denote the set of all intial configuratios and those that can be generated in all executions of P when the events are those in S
- Let S be continuous, j-adjacency preserving and F-admissible, $F > 0$. Let P be a $(\lfloor (n-1)/2 \rfloor + 2)$-agreement protocol. If $\mathcal{P}(P, S)$ contains two accessible l-adjacent cofigurations, a 0-valent and a 1-valent one, then P is not correct in spite of F communication faults in S

A Theorem to Help with the Proof (2/3)

- Proof shortly:
  - First we make a contradiction that P is correct. Then if we let A and B be two j-adjacent accessible configurations that are 0-valent and 1-valent respectively
  - Now because S is j-adjacency preserving there exsists events for both A and B such that the resulting configurations are j-adjacent. We can continue reasoning this way further
  - As P is correct there exists a time $t \geq 1$ such that both configurations A and B have reached a decision value
  - As A is 0-valent (B is 1-valent), at least $\lceil \frac{n}{2} \rceil + 1$ entities have value 0 (value 1 with B)

A Theorem to Help with the Proof (3/3)

- Proof shortly (continued):
  - This means that is at least one entity x_i, $i \neq j$ that has value 0 in configuration resulting from A and 1 in configuration resulting from B
  - However since the resulting configurations are j-adjacent, they only differ in the state of one entity x_j: a contradiction
  - P is not correct

Impossibility of Strong Majority (1/3)

- Theorem: Let S be adjacency preserving, continuous and F-admissible. Then no k-agreement protocol is correct in spite of F communication faults in S for $k > \lceil n/2 \rceil$
- Proof: Assume P is a correct $(\lceil n/2 \rceil + 1)$-agreement protocol in spite of F communication faults when the message system returns only events in S. The proof involves 2 steps:
  - First, it is argued that there exists some bivalent initial configuration
  - Second, it is shown that entering a configuration with decision value can be postponed forever

Impossibility of Strong Majority (2/3)

- Lemma: $\mathcal{P}(P, S)$ has an intial bivalent configuration
- Proof: Let every configuration in $\mathcal{P}(P, S)$ be 0-valent or 1-valent and let P be correct
  - By definition there exists at least one of both 0-valent and 1-valent configurations
  - Then there must be a 0-valent and 1-valent initial configurations that are adjacent
  - By the earlier theorem it follows that P is not correct
  - It follows that there must be a bivalent initial configuration

Impossibility of Strong Majority (3/3)

- Lemma: Every bivalent configuration in $\mathcal{P}(P, S)$ has a succeeding bivalent configuration
- Proof: Let C be a bivalent configuration in $\mathcal{P}(P, S)$
  - If C has no bivalent configuration then C has at least 0-valent and 1-valent succeeding configurations, say A and B
  - Because S is continuous there exists a sequence of events that make configurations A and B adjacent.
  - By the earlier theorem it follows that P is not correct
  - It follows that every bivalent configuration has a succeeding bivalent configuration

Consequences

- The impossibility result offers a powerful tool for proving impossibilty results for nontrivial agreement
- No nontrivial agreement is possible for the faults of set S of events, if it can be shown that S is
  - Adjacency preserving
  - Continuous
  - F-admissible

Outline

Additional Assumptions

- 1. Connectivity
- 2. Synch
- 3. All entities start simultaneously
- 4. Each entity has a map of the network

Reaching Unanimity

- The conditions for reaching unanimity depend on the type and number of faults and also on the edge connectivity $C_{edge}(G)$ of G
- In all cases, we will reach unanimity, in spite of F communication faults per clock cycle by computing the OR of the input values and deciding on that value
- To compute the OR we need a reliable broadcasting method that will complete within a fixed amount of time T (also called timeout value)
- The broadcast mechanism will differ depending on the nature of the faults present in the system

Single Type Faults: Omissions

- We have seen earlier that broaadcast is impossible if $F \geq c_{edge}(G)$ (Lemma 7.1.1)
- We can broadcast if $F \leq c_{edge}(G) - 1$
- Algorithm Bcast-Omit
  - 1. To broadcast in G, node x sends its message at time 0 and continues transmitting it to all its neigbours until time $T(G) - 1$
  - 2. A node receiving the message at time $t < T(G)$ will transmit the message to all its other neigbours until time $T(G) - 1$
  - Where $T(G) \leq c_{edge}(G)n - 2c_{edge}(G) + 1$
  - B(Bcast-omit) $\leq 2m(G)T(G)$

Single Type Faults: Additions

- To deal with additions in fully synchronous system is possible but eexpensive
- If every entity transmits on every clock cycle in leaves no room for additions
- Implementation
  - Every entity transmits for the first $T(G) - 1$ time units
  - Intially each entity transmits it's own value
  - If at any time entity ís aware of a 1 in the system it starts transmitting it
- Unanimity can be reached regardless of the number of faults in time $T = diam(G)$ transmitting $2m(G)diam(G)$ bits

Single Type Faults: Corruptions

- Dealing with corruptions is easy; Because no omissions or additions can occur, if a node starts a broadcast every other node will receive a message (possibly corrupted)
- Only nodes with intial value 1 will start broadcast and the content of the messages is not regarded
- Unanimity can be reached regardless of the number of faults in time $T = diam(G)$ and transmitting at most $2m(G)$ bits

Composite Faults

- Omissions and corruptions
  - The situation is fortunately no worse than system with only omission faults
- Omissions and additions
  - We can start with the same idea we used with additions only. However the omissions can stop the nodes from receiving broadcast so we will have a the same limit to the number of faults than with a system that has omission faults only
- Additions and corruptions
  - The computation of OR is quite difficult. We need to define few techniques to help us reach unanimity.

Time Splice

- We distinguish between even and odd clock ticks; an even clock tick and successive odd clock tick constitute a communication cycle
- To broadcast 0 (1), x will send a message to all its neighbours only on even clock ticks (odd)
- When receiving a message at an even (odd) clock tick entity *y* will forward it only on even (odd) clock ticks
- This technique does not solve the problem created by additions

Reliable Neighbour Transmission (1/2)

- Cosider an entity x and its neighbour y. Let SP(x, y) be the set of $c_{edge}(G)$ shortest disjoint paths form x to y To communicate a message from x to y, the message is sent by x simultaneously to all paths in SP(x, y)
- Algorithm
  - For each neighbouring pair x, y and paths SP(x, y), every entity determines in which of these paths it resides
  - To send message M and information about the path to neighbour y, x will send along each path in SP(x, y) for t consecutive communication cycles
  - An entity z along the path, upon receiving in communication cycle k a message for y with correct path information will forward it for $t - k$ cycles

Reliable Neighbour Transmission (2/2)

- Incorrect path information and incorrect timing are detectable
- Properties:
  - In t communication cycles, at most Ft copies of incorrect messages arrive at y
  - y will receive at least $(l-1) + c_{edge}(G)(t-(l-1))$ copies (possibly corrupted of the bit from x within $t > l$ communication cyclesM; where l is longest of the paths in $SP(x, y)$ for any neighbouring x, y
  - To make it possible for y to determine the original bit sent by x it is sufficent that $t > (c(G)-1)(l-1)$

Addition and Corruption faults

- Let us combine these two technigues. That is all entities broadcast their initial value using the time splice technique. However each step of the broadcast, in which every involved entity sends the bit to its neighbours is done using the reliable neighbour transmission
- This means that every step of the broadcast takes now t communication cycles
- Consider that the broadcast requires $diam(G)$ steps, hence it is possible to compute the OR in spite of $c_{edge}(G) - 1$ additions and corruptions in time at most $2diam(G)(c_{edge}(G) - 1)(l - 1)$ and the number of bits is at most $4m(G)(c_{edge}(G) - 1)(l - 1)$

Byzantine faults (1/2)

- We will use reliable neighbour transmission without time splice technique to reach unanimity in byzantine case
- We will need to define a decision process for y to determine the correct bit
- Acceptance rule: y selects the as correct the bit value received most often during the t time units

Byzantine faults (2/2)

- Why this works with $F \leq (\lceil c_{edge}(G)/2 \rceil - 1)$:
  - Let us pretend that no faults occur; then on the first $(l-1)$ clock cycles a message will reach y After that a message will reach y from every path in SP(x, y)
  - Thus at least $n = (l-1) + c_{edge}(G)(t-(l-1))$ messages will reach y
  - With at most tF faults per cycle the minimum number of correct messages is the difference $n - tF$. Now for the acceptance rule to work correctly we need that the number of correct messages is larger than the number of faulty messages
  - This is satisfied by $t > (c_{edge}(G) - 1)(l-1))$
- Because broadcast requires $diam(G)$ ticks, unanimity can be reached at time $T \leq diam(G)(c_{edge}(G) - 1)(l-1)$