Certification of Large Distributed Computations with Task Dependencies in Hostile Environments

Thierry Gautier, Samir Jafar, Axel Krings, Franck Leprévost, Jean-Louis Roch, Sébastien Varrette

Equipe MOAIS Laboratoire ID-IMAG, France

Jean-Louis.Roch@imag.fr

Presentation Outline

- Motivation: Application and Threat
- Execution Model
- Certification with independent tasks
- Certification with task dependencies
- Results
- Conclusions and Future Work
Target Application

- Large-Scale Global Computing Systems

- Subject Application to Dependability Problems
  - Can be addressed in the design

- Subject Application to Security Problems
  - Requires solutions from the area of survivability, security, fault-tolerance
Global Computing Architecture

- Large-scale distributed systems (e.g. Grid, P2P)
- Transparent allocation of resources
Unbounded Environments

- In the Survivability Community our general computing environment is referred to as

  Unbounded Environment

  - Lack of physical / logical bound
  - Lack of global administrative view of the system.

What risks are we subjecting our applications to?
Typical Application

- Computation intensive parallel application
  - Medical (mammography comparison)
Two kinds of failures (1/2)

1. Node failures
   - “fail stop” model
Unreliability in the absence of Fault Tolerance Mechanism

- Computation on Cluster
  - MTBF = 2000 days (48,000h, approx. 5 1/2 years)
  - Unreliability of one node: \( F(t) = 1 - R(t) = 1 - e^{-\lambda t} \)
Fault Tolerance Approaches

- Simplified Taxonomy for Fault Tolerance Protocols

<table>
<thead>
<tr>
<th>FT Protocol</th>
<th>Duplication</th>
<th>Checkpointing</th>
<th>Message-Logging</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Uncoordinated</td>
<td>Coordinated</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Communication-induced</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Rely on a “stable storage”
  - persistent and assumed to be reliable [Kaapi / Athapascan ]
  - If not persistent: only duplication of saved data (checkpoint / message)
    » probabilistic FT protocols: fault tolerance is guaranteed with good probability
Two kinds of failures (2/2)

2. Task forgery
   - “massive attacks”
How bad is the Problem?

- Vulnerabilities reported (CERT/CC statistics)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Vulnerabilities</td>
<td>171</td>
<td>345</td>
<td>311</td>
<td>262</td>
<td>417</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>2000</th>
<th>2001</th>
<th>2002</th>
<th>2003</th>
<th>2004</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vulnerabilities</td>
<td>1,090</td>
<td>2,437</td>
<td>4,129</td>
<td>3,784</td>
<td>3,780</td>
</tr>
</tbody>
</table>

Total vulnerabilities reported (1995-2004): 16,726
How bad is the Problem?

- Incidents reported (CERT/CC statistics)

<table>
<thead>
<tr>
<th>Year</th>
<th>1988</th>
<th>1989</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incidents</td>
<td>6</td>
<td>132</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Incidents</td>
<td>252</td>
<td>406</td>
<td>773</td>
<td>1,334</td>
<td>2,340</td>
<td>2,412</td>
<td>2,573</td>
<td>2,134</td>
<td>3,734</td>
<td>9,859</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>2000</th>
<th>2001</th>
<th>2002</th>
<th>2003</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incidents</td>
<td>21,756</td>
<td>52,658</td>
<td>82,094</td>
<td>137,529</td>
</tr>
</tbody>
</table>

Total incidents reported (1988-2003): 319,992
Fault Models

- Simplified Fault Taxonomy

- Fault-Behavior and Assumptions
  - Independence of faults
  - Common mode faults -> towards arbitrary faults!

- Fault Sources
  - Trojan, virus, DOS, etc.
  - How do faults affect the overall system?
Assumptions

- Anything is possible!
  » and it will happen!

- Malicious act will occur sooner or later

- It is hard or impossible to predict the behavior of an attack
Attacks and their impact

- **Attacks**
  - single nodes, difficult to solve with certification strategies
  - solutions: e.g. intrusion detection systems (IDS)

- **Massive Attacks**
  - affects large number of nodes
  - may spread fast (worm, virus)
  - may be coordinated (Trojan)

- **Impact of Attacks**
  - attacks are likely to be widespread within neighborhood, e.g. subnet

- **Our focus: massive attacks**
  - virus, trojan, DoS, etc.
Certification Against Attacks

- Mainly addressed for **independent tasks**

- **Current approaches**
  - Simple checker [Blum97]
  - Voting [SETI@home]
  - Spot-checking [Germain-Playez 2003, based on Wald test]
  - Blacklisting
  - Credibility-based fault-tolerance [Sarmenta 2003]
  - Partial execution on reliable resources (partitioning) [Gao-Malewicz 2004]
  - Re-execution on reliable resources

- Certification of Computation
**Presentation Outline**

- Motivation: Application and Threat
- **Execution Model**
- Certification with independent tasks
- Certification with task dependencies
- Results
- Conclusions and Future Work
Definitions and Assumptions

- **Dataflow Graph**
  - \( G = (V, E) \)
    - \( V \) finite set of vertices \( v_i \)
    - \( E \) set of edges \( e_{jk} \) vertices \( v_j, v_k \in V \)

- **Two kinds of tasks**
  - \( T_i \) Tasks in the traditional sense
  - \( D_j \) Data tasks inputs and outputs
Global Computing Platform (GCP)

- GCP includes workers, checkpoint server and verifiers
Definitions

- Executions in **unreliable** environment
  
  $E$ execution of workload represented by $G$
  
  $i(T,E)$ input to $T$ in execution $E$
  
  $o(T,E)$ output of $T$ in execution $E$

- Executions in **reliable** environment: Verifier
  
  $\hat{E}$ execution of workload $G$ on Verifier
  
  $i(T,\hat{E})$ input to $T$ in execution $\hat{E}$
  
  $\hat{o}(T,\hat{E})$ output of $T$ in execution $\hat{E}$
  
  $\hat{o}(T,E)$ output of $T$ with input from $E$ executing on verifier

Note: notations $\hat{o}(T,\hat{E})$ and $\hat{o}(T,E)$ differ!

- If $E = \hat{E}$ then $E$ is said to be “correct”
  otherwise $E$ is said to have “failed”
Monte Carlo certification: (analogy to Miller-Rabin)
- a randomized algorithm that
  1. takes as input $E$ and an arbitrary $\varepsilon$, $0 < \varepsilon \leq 1$
  2. delivers
     - either CORRECT
     - or FAILED, together with a proof that $E$ has failed

- certification is with error $\varepsilon$ if the probability of answer CORRECT, when $E$ has actually failed, is less than or equal to $\varepsilon$. 
What does the certification really mean?
- what is the real interpretation of $E = \hat{E}$
- connection between $E = \hat{E}$ and massive attack
- use $E \neq \hat{E}$ as a “tool” to determine if a massive attack has occurred

Monte Carlo certification against massive attacks
- number of tasks actually failed/attacked $n_F$
- consider two scenarios
  » $n_F = 0$
  » $n_F$ is large => massive attack

Attack Ratio $q$

$$n_q = \left\lceil nq \right\rceil \leq n_F$$
Monte Carlo Test

- Algorithm MCT
  1. Uniformly select one task $T$ in $G$
     we know input $i(T,E)$ and output $o(T,E)$ of $T$ from checkpoint server
  2. Re-execute $T$ on verifier, using $i(T,E)$ as inputs, to get output $\hat{o}(T,E)$
     If $o(T,E) \neq \hat{o}(T,E)$ return FAILED
  3. Return CORRECT
Certification of Independent Tasks

How many independent executions of MCT are necessary to achieve certification of $E$ with probability of error $\leq \varepsilon$?

$$N \geq \left\lceil \frac{\log \varepsilon}{\log(1 - q)} \right\rceil$$

- Prob. that MCT selects a non-forged tasks is $$\frac{n - n_F}{n} \leq 1 - q$$

- $N$ independent applications of MCT results in $$\varepsilon \leq (1 - q)^N$$
Certification of Independent Tasks

- Relationship between attack ratio and N
Certification of Independent Tasks

- Relationship between certification error and N

For $q = 1\%$:
- $300$ checks $\Rightarrow \varepsilon < 5\%$
- $4611$ checks $\Rightarrow \varepsilon < 10^{-20}$
- $24000$ checks $\Rightarrow \varepsilon < 10^{-125}$
Presentation Outline

- Motivation: Application and Threat
- Execution Model
- Certification with independent tasks
- Certification with task dependencies
- Results
- Conclusions and Future Work
Certification and Task Dependencies

- What does a re-execution really tell us w.r.t. the result?
  - One can only talk about outputs of tasks, not tasks!
  - If $o(T,E) \neq \hat{o}(T,E)$ we know that an error has occurred
  - If $o(T,E) = \hat{o}(T,E)$ we cannot say much at all!
    » for independent tasks this indicated a good task/result
    » what do we know about the inputs?
      ■ in the presence of error propagation -- not much!
    » if the verifier uses $i(T,\hat{E})$ then $o(T,E) = \hat{o}(T,\hat{E})$ indicates a good result
      but we don’t have $\hat{E}$, (would require total re-execution on verifier)
Certification and Task Dependencies

- The concept of “Initiator”
  - $o(T,E) = \hat{o}(T,E)$ is only useful if we know that the inputs are correct
    » this implies that $T$ has no forged predecessors

- Definition:
  An initiator is a falsified tasks that has no falsified predecessors

- Worst case assumption is very conservative
  » one still might detect a falsified non-initiator
  » but there is no guarantee
Certification and Task Dependencies

- Certification is now based on initiators
- Using Algorithm MCT we get

\[ N \geq \frac{\log \varepsilon}{\log(1 - \frac{n_I}{n})} \]
Certification and Task Dependencies

\( G^\leq(V) \) predecessor graph of all tasks in \( V \)

\( k \leq n_F \) be the number of falsified tasks assumed

\( I(F) \) set of all initiators

- Minimum Number of Initiators

\[ \gamma_V(k) = \min | G^\leq(V) \cap I(F) | \]

- Minimal Initiator Ratio

\[ \Gamma_V(k) = \frac{\gamma_V(k)}{| G^\leq(V) |} \]
Algorithm EMCT

1. Uniformly select one task $T$ in $G$

2. Re-execute all $T_j$ in $G^{\leq}(T)$, which have not been verified yet, with input $i(T,E)$ on a verifier and return FAILED if for any $T_j$ we have $o(T_j,E) \neq \hat{o}(T_j,E)$

3. Return CORRECT

Behavior

- disadvantage: the entire predecessor graph needs to be re-executed
- however: the cost depends on the graph
  - luckily our application graphs are mainly trees
Results of independent tasks still hold,
  - but $N$ hides the cost of verification
    » independent tasks: $C = 1$
    » dependent tasks: $C = |G^<(T)|$
Reducing the cost of verification

For EMCT the entire predecessor graph had to be verified
To reduce verification cost two approaches are considered next:

1. Verification with fractions of $G \leq (T)$
2. Verification with fixed number of tasks in $G \leq (T)$
Verifying with fractions of $G^{\leq}(T)$

- Algorithm EMCT$\alpha(E)$

1. Uniformly choose one task $T$ in $G$.

2. Uniformly select $n_\alpha = \lceil \alpha |G^{\leq}(T)| \rceil$ tasks in $G^{\leq}(T)$ and let this set be denoted by $A$. If for any $T_j \in A$, that has not been verified yet, re-execution on a verifier results in $\hat{o}(T_j, E) \neq o(T_j, E)$ then return FAILED.

3. Return CORRECT.
Verifying with fractions of $G^\leq(T)$

For Algorithm $EMCT_\alpha(E)$

**Lemma 1** Let $T$ be a task randomly chosen by $EMCT_\alpha(E)$. Then the probability of error, $e_\alpha$, when $EMCT_\alpha(E)$ returns CORRECT is given by

$$e_\alpha \leq \begin{cases} 
(1 - q\alpha\Gamma_T(n_q)) & \text{for } 0 < \alpha \leq 1 - \Gamma_T(n_q) \\
(1 - q) & \text{otherwise}.
\end{cases}$$
Verifying with fractions of $G^{\leq}(T)$

- For Algorithm EMCT$\alpha(E)$

**Theorem 1** Let $E$ be an execution with dependencies that is either correct or massively attacked with ratio $q$. Given $\epsilon$ and $0 < \alpha \leq 1$, $N$ independent invocations of Algorithm EMCT$\alpha(E)$ provide a certification with error probability

$$
\epsilon \leq \begin{cases} 
(1 - q\alpha \Gamma_G(n_q))^N & \text{for } 0 < \alpha \leq 1 - \Gamma_T(n_q) \\
(1 - q)^N & \text{otherwise.}
\end{cases}
$$
Verifying fixed numbers of tasks

- We will now modify algorithm EMCT so that only a fixed number of tasks in the predecessors are verified.
  - We limit our investigations to unity, i.e. one task is verified.
Verifying fixed numbers of tasks

- Algorithm EMCT\(^1\)(E)

1. Uniformly choose one task \(T\) in \(G\).

2. Uniformly select a single \(T_j\) in \(G^\leq(T)\). If re-execution of \(T_j\) on a verifier results in \(\hat{o}(T_j, E) \neq o(T_j, E)\) then return FAILED.

3. Return CORRECT.
Verifying fixed numbers of tasks

- For Algorithm EMCT$^1(E)$

Lemma 2 Let $T$ be a task randomly chosen by $EMCT^1(E)$ and let $V = G^\leq(T)$. Then the probability of error, $e_1$, when $EMCT^1(E)$ returns CORRECT is given by

$$e_1 \leq 1 - \frac{n_F}{n} \Gamma_T(n_F) \leq 1 - q \Gamma_T(n_q)$$
Verifying fixed numbers of tasks

- For Algorithm EMCT\(^1\)(E)

**Theorem 2** Let \( E \) be an execution with dependencies that is either correct or massively attacked with ratio \( q \). Given \( \epsilon \) then \( N \) independent invocations of Algorithm EMCT\(^1\)(E) provide a certification with error probability

\[
\epsilon \leq (1 - q \Gamma_G(n_q))^N.
\]
The cost of certification

◆ A balance between \( N \) and \( C \)

◆ Monte Carlo certification for a given \( \varepsilon \):
  1. a priori convergence
     - determine up front how many times one has to verify
     - one does not know which tasks are selected
  2. run-time convergence
     - run until certain \( \varepsilon \) is achieved
     - take advantage of knowledge about task selected
  3. for general graphs
  4. for special graphs (e.g. out-trees)

Note: For independent tasks a priori and run-time convergence are the same.
Results for pathological cases

- Number of effective initiators
  - this is the # of initiators as perceived by the algorithm
  - e.g. for EMCT an initiator in $G^\leq(T)$ is always found, if it exists

<table>
<thead>
<tr>
<th></th>
<th>$MCT(E)$ [7]</th>
<th>$EMCT(E)$ [7]</th>
<th>$EMCT_\alpha(E)$</th>
<th>$EMCT^1(E)$</th>
</tr>
</thead>
<tbody>
<tr>
<td># of effective initiators</td>
<td>$\left\lceil \frac{n_q}{1-d} \right\rceil$</td>
<td>$n_q$</td>
<td>$n_q\alpha\Gamma_T(n_q)$ or $n_q$</td>
<td>$n_q\Gamma_T(n_q)$</td>
</tr>
<tr>
<td>Probability of error</td>
<td>$1 - \frac{\log \epsilon}{n}$</td>
<td>$1 - q$</td>
<td>$1 - q\alpha\Gamma_T(n_q)$ or $1 - q$</td>
<td>$1 - q\Gamma_T(n_q)$</td>
</tr>
<tr>
<td>A priori convergence</td>
<td>$\left\lceil \frac{n_q}{1-d} \right\rceil \log(1-q)$</td>
<td>$\frac{\log \epsilon}{\log(1-q)}$</td>
<td>$\frac{\log \epsilon}{\log(1-q\alpha\Gamma_G(n_q))}$ or $\frac{\log \epsilon}{\log(1-q)}$</td>
<td>$\frac{\log \epsilon}{\log(1-q\Gamma_G(n_q))}$</td>
</tr>
<tr>
<td>$q_e$ a priori</td>
<td>$\left\lceil \frac{n_q}{1-d} \right\rceil$</td>
<td>$q$</td>
<td>$q\alpha\Gamma_G(n_q)$ or $q$</td>
<td>$q\Gamma_G(n_q)$</td>
</tr>
<tr>
<td>$q_e$ run-time</td>
<td>$\left\lceil \frac{n_q}{1-d} \right\rceil$</td>
<td>$q$</td>
<td>$q\alpha\Gamma_T(n_q)$ or $q$</td>
<td>$q\Gamma_T(n_q)$</td>
</tr>
<tr>
<td>Verification cost (exact)</td>
<td>1</td>
<td>$</td>
<td>G^\leq(T)</td>
<td>$</td>
</tr>
<tr>
<td>Max. cost (out-tree)</td>
<td>1</td>
<td>$h$</td>
<td>$\alpha h$</td>
<td>1</td>
</tr>
</tbody>
</table>
Results for pathological cases

- Probability of error induced by one invocation
  - derived for each algorithm

<table>
<thead>
<tr>
<th></th>
<th>$MCT(E)$ [7]</th>
<th>$EMCT(E)$ [7]</th>
<th>$EMCT_\alpha(E)$</th>
<th>$EMCT_1(E)$</th>
</tr>
</thead>
<tbody>
<tr>
<td># of effective initiators</td>
<td>$\left\lceil \frac{n q}{\ln(1-d)} \right\rceil$</td>
<td>$n_q$</td>
<td>$n_q \alpha \Gamma_T(n_q)$ or $n_q$</td>
<td>$n_q \Gamma_T(n_q)$</td>
</tr>
<tr>
<td>Probability of error</td>
<td>$1 - \frac{n_q}{\ln(1-d)}$</td>
<td>$1 - q$</td>
<td>$1 - q \alpha \Gamma_T(n_q)$ or $1 - q$</td>
<td>$1 - q \Gamma_T(n_q)$</td>
</tr>
<tr>
<td>A priori convergence</td>
<td>$\frac{\ln(1-q)}{\ln(1-q\alpha \Gamma_G(n_q))}$</td>
<td>$\frac{\ln(1-q)}{\ln(1-q\alpha \Gamma_G(n_q))}$</td>
<td>$\frac{\ln(1-q)}{\ln(1-q\alpha \Gamma_G(n_q))}$</td>
<td></td>
</tr>
<tr>
<td>$q_e$ a priori</td>
<td>$\left\lceil \frac{n q}{\ln(1-d)} \right\rceil$</td>
<td>$q$</td>
<td>$q \alpha \Gamma_G(n_q)$ or $q$</td>
<td>$q \Gamma_G(n_q)$</td>
</tr>
<tr>
<td>$q_e$ run-time</td>
<td>$\left\lceil \frac{n q}{\ln(1-d)} \right\rceil$</td>
<td>$q$</td>
<td>$q \alpha \Gamma_T(n_q)$ or $q$</td>
<td>$q \Gamma_T(n_q)$</td>
</tr>
<tr>
<td>Verification cost (exact)</td>
<td>$1$</td>
<td>$</td>
<td>G^{\leq}(T)</td>
<td>$</td>
</tr>
<tr>
<td>Max. cost (out-tree)</td>
<td>$1$</td>
<td>$h$</td>
<td>$\alpha h$</td>
<td>$1$</td>
</tr>
</tbody>
</table>
Results for pathological cases

- A priori convergence ($N$ is determined a priori)
  - cannot take advantage of run-time knowledge
  - has to use $\Gamma_G(n_q)$ rather than $\Gamma_T(n_q)$
  - $q_e$ is the effective attack ratio

$$N \approx \left\lceil \frac{\log \varepsilon}{\log(1 - q_e)} \right\rceil$$

<table>
<thead>
<tr>
<th></th>
<th>$MCT(E)$ [7]</th>
<th>$EMCT(E)$ [7]</th>
<th>$EMCT_\alpha(E)$</th>
<th>$EMCT^1(E)$</th>
</tr>
</thead>
<tbody>
<tr>
<td># of effective initiators</td>
<td>$\left\lceil \frac{n_q}{1 - q_e} \right\rceil$</td>
<td>$n_q$</td>
<td>$n_q \alpha \Gamma_T(n_q)$ or $n_q$</td>
<td>$n_q \Gamma_T(n_q)$</td>
</tr>
<tr>
<td>Probability of error</td>
<td>$1 - \frac{\log \varepsilon}{n_q}$</td>
<td>$1 - q$</td>
<td>$1 - q \alpha \Gamma_T(n_q)$ or $1 - q$</td>
<td>$1 - q \Gamma_T(n_q)$</td>
</tr>
<tr>
<td>A priori convergence</td>
<td>$\frac{\log \varepsilon}{\log(1 - \frac{n_q}{1 - q_e})}$</td>
<td>$\frac{\log \varepsilon}{\log(1 - q)}$</td>
<td>$\frac{\log \varepsilon}{\log(1 - q \alpha \Gamma_G(n_q))}$ or $\frac{\log \varepsilon}{\log(1 - q)}$</td>
<td>$\frac{\log \varepsilon}{\log(1 - q \Gamma_G(n_q))}$</td>
</tr>
<tr>
<td>$q_e$ a priori</td>
<td>$\left\lceil \frac{n_q}{1 - q_e} \right\rceil$</td>
<td>$q$</td>
<td>$q \alpha \Gamma_G(n_q)$ or $q$</td>
<td>$q \Gamma_G(n_q)$</td>
</tr>
<tr>
<td>$q_e$ run-time</td>
<td>$\left\lceil \frac{n_q}{1 - q_e} \right\rceil$</td>
<td>$q$</td>
<td>$q \alpha \Gamma_T(n_q)$ or $q$</td>
<td>$q \Gamma_T(n_q)$</td>
</tr>
<tr>
<td>Verification cost (exact)</td>
<td>1</td>
<td>$</td>
<td>G^{\leq}(T)</td>
<td>$</td>
</tr>
<tr>
<td>Max. cost (out-tree)</td>
<td>1</td>
<td>$h$</td>
<td>$\alpha h$</td>
<td>1</td>
</tr>
</tbody>
</table>
## Results for pathological cases

- **Run-time convergence** ($N$ is determined at run-time)
  - takes advantage of run-time knowledge
  - initial verification $\varepsilon_e = 1 - q_e$
  - each verification $\varepsilon_e = \varepsilon_e (1 - q_e)$
  - until $\varepsilon_e \leq \varepsilon$

\[
N \geq \left\lfloor \frac{\log \varepsilon}{\log(1 - q_e)} \right\rfloor
\]

### Results for pathological cases

<table>
<thead>
<tr>
<th></th>
<th>$MCT(E)$ [7]</th>
<th>$EMCT(E)$ [7]</th>
<th>$EMCT_\alpha(E)$</th>
<th>$EMCT_1(E)$</th>
</tr>
</thead>
<tbody>
<tr>
<td># of effective initiators</td>
<td>( \left\lfloor \frac{n_q}{1 - \frac{q}{1 - d}} \right\rfloor )</td>
<td>( n_q )</td>
<td>( n_q \alpha \Gamma_T(n_q) ) or ( n_q )</td>
<td>( n_q \Gamma_T(n_q) )</td>
</tr>
<tr>
<td>Probability of error</td>
<td>( 1 - \frac{n_q}{\log(1 - q)} )</td>
<td>( 1 - q )</td>
<td>( 1 - q \alpha \Gamma_T(n_q) ) or ( 1 - q )</td>
<td>( 1 - q \Gamma_T(n_q) )</td>
</tr>
<tr>
<td>A priori convergence</td>
<td>( \frac{\log \varepsilon}{\log(1 - q)} )</td>
<td>( \frac{\log \varepsilon}{\log(1 - q)} )</td>
<td>( \frac{\log \varepsilon}{\log(1 - q \alpha \Gamma_G(n_q))} ) or ( \frac{\log \varepsilon}{\log(1 - q \Gamma_G(n_q))} )</td>
<td>( \frac{\log \varepsilon}{\log(1 - q \Gamma_G(n_q))} )</td>
</tr>
<tr>
<td>$q_e$ a priori</td>
<td>( \left\lfloor \frac{n_q}{1 - \frac{q}{1 - d}} \right\rfloor )</td>
<td>( q )</td>
<td>( q \alpha \Gamma_G(n_q) ) or ( q )</td>
<td>( q \Gamma_G(n_q) )</td>
</tr>
<tr>
<td>$q_e$ run-time</td>
<td>( \left\lfloor \frac{n_q}{1 - \frac{q}{1 - d}} \right\rfloor )</td>
<td>( q )</td>
<td>( q \alpha \Gamma_T(n_q) ) or ( q )</td>
<td>( q \Gamma_T(n_q) )</td>
</tr>
<tr>
<td>Verification cost (exact)</td>
<td>( 1 )</td>
<td>(</td>
<td>G_{\leq}(T)</td>
<td>)</td>
</tr>
<tr>
<td>Max. cost (out-tree)</td>
<td>( 1 )</td>
<td>( h )</td>
<td>( \alpha h )</td>
<td>( 1 )</td>
</tr>
</tbody>
</table>
Results for pathological cases

- Verification cost
  - per invocation of the algorithm
  - special case: out-tree

<table>
<thead>
<tr>
<th></th>
<th>$MCT(E)$ [7]</th>
<th>$EMCT(E)$ [7]</th>
<th>$EMCT_\alpha(E)$</th>
<th>$EMCT^1(E)$</th>
</tr>
</thead>
<tbody>
<tr>
<td># of effective initiators</td>
<td>$\left\lceil \frac{n_q}{1-d^h} \right\rceil$</td>
<td>$n_q$</td>
<td>$n_q\alpha \Gamma_T(n_q)$ or $n_q$</td>
<td>$n_q \Gamma_T(n_q)$</td>
</tr>
<tr>
<td>Probability of error</td>
<td>$1 - \frac{n_q}{n}$</td>
<td>$1 - q$</td>
<td>$1 - q\alpha \Gamma_T(n_q)$ or $1 - q$</td>
<td>$1 - q \Gamma_T(n_q)$</td>
</tr>
<tr>
<td>A priori convergence</td>
<td>$\frac{\log \epsilon}{\log(1-q)} \left\lceil \frac{n_q}{1-d^h} \right\rceil$</td>
<td>$\frac{\log \epsilon}{\log(1-q)}$</td>
<td>$\frac{\log \epsilon}{\log(1-q)} \left\lceil \frac{n_q}{1-d^h} \right\rceil$</td>
<td>$\frac{\log \epsilon}{\log(1-q \Gamma_G(n_q))}$</td>
</tr>
<tr>
<td>$q_e$ a priori</td>
<td>$\left\lceil \frac{n_q}{1-d^h} \right\rceil$</td>
<td>$q$</td>
<td>$q\alpha \Gamma_G(n_q)$ or $q$</td>
<td>$q \Gamma_G(n_q)$</td>
</tr>
<tr>
<td>$q_e$ run-time</td>
<td>$\left\lceil \frac{n_q}{1-d} \right\rceil$</td>
<td>$q$</td>
<td>$q\alpha \Gamma_T(n_q)$ or $q$</td>
<td>$q \Gamma_T(n_q)$</td>
</tr>
<tr>
<td>Verification cost (exact)</td>
<td>1</td>
<td>$</td>
<td>G^\leq(T)</td>
<td>$</td>
</tr>
<tr>
<td>Max. cost (out-tree)</td>
<td>1</td>
<td>$h$</td>
<td>$\alpha h$</td>
<td>1</td>
</tr>
</tbody>
</table>
Conclusions

- Certification of large distributed applications
  - hostile environments with no assumptions on fault model

- Considered task dependencies
  - tasks or data may be manipulated
  - allows for error propagation (much more difficult than independent case)
  - very difficult to speculate on the behavior of a falsified task

- Several probabilistic certification algorithms were introduced
  - based on re-execution on verifier (reliable resource)
  - inputs available from dataflow checkpoints

- Certification:
  - very low probability of error can be achieved
  - number of tasks to verify is relatively small, depending on graph
  - relationship between attack rate and probability of error
Questions?
The impact of graph $G$

- Knowing the graph, an attacker may attempt to minimize the visibility of even a massive attack with ration $q$.
- What is the number of initiators one might have to expect in a graph?
  » In the worst case we have

$$
\gamma_G(n_F) = \left[ \frac{n_F}{1 - d^h} \right] \left[ \frac{1}{1 - d} \right]
$$
### Results for MCT and EMTC

- **Considered**
  - General graphs
  - Out-trees (application domain based on out/in-trees)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>MCT</th>
<th>EMCT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of effective initiators</td>
<td>[ \left\lfloor \frac{n_q}{1-d^h} \right\rfloor ] ( \frac{1-d^h}{1-d} )</td>
<td>( n_q )</td>
</tr>
<tr>
<td>Probability of error</td>
<td>1 - [ \frac{\left(1-d^h\right)^q}{\left(1-d\right)^n} ]</td>
<td>1 - ( q )</td>
</tr>
<tr>
<td>Verification cost: general ( G )</td>
<td>1</td>
<td>( O(n) )</td>
</tr>
<tr>
<td>Verification cost: ( G ) is out-tree</td>
<td>1</td>
<td>( h \log_d(n_v) )</td>
</tr>
<tr>
<td>Ave. # effective initiators, ( G ) is out-tree</td>
<td>[ \left\lfloor \frac{n_q}{\left(1-(h+2)d^h+1+(h+1)d^h+2\right)} \right\rfloor ] ( \frac{1-(h+2)d^h+1+(h+1)d^h+2}{(1-d)(1-d^h+1)} )</td>
<td>( n_q )</td>
</tr>
</tbody>
</table>
Relationship between quantities

Given a subset $V$ of tasks in $G$.

What are the relationships between $\gamma_V(k)$, $\gamma_G(k)$ and $n_I$ with respect to $k = n_q$ or $k = n_F$?

By definition

$q \leq n_F / n$ and thus $n_q \leq n_F$

also

$n_I \leq n_F$
Relationship between quantities

- With respect to $n_F$ we always have
  \[ \gamma_V(n_F) \leq \gamma_G(n_F) \leq n_I \leq n_F \]
  - But where does $n_q$ fit into this inequality?
  - The only certain relationship is $n_q \leq n_F$

- With respect to $n_q$ we always have
  \[ \gamma_V(n_q) \leq \gamma_G(n_q) \leq n_q \leq n_F \]
  - But where does $n_I$ fit into this inequality?
  - The only certain relationship is $\gamma_G(n_q) \leq n_I \leq n_F$
With respect to $n_q \leq n_F$ we can compare directly

\[
\begin{align*}
\gamma_V(n_q) &\leq \gamma_V(n_F) \\
\gamma_G(n_q) &\leq \gamma_G(n_F)
\end{align*}
\]

Thus

\[
\begin{align*}
\Gamma_V(n_q) &\leq \Gamma_V(n_F) \\
\Gamma_G(n_q) &\leq \Gamma_G(n_F)
\end{align*}
\]