

Distributed Systems

ALI KAMANDI, PH.D.

SCHOOL OF ENGINEERING SCIENCE

COLLEGE OF ENGINEERING

UNIVERSITY OF TEHRAN

KAMANDI@UT.AC.IR

مدل های خطا

مدل خطا مشخص می کند که به چه طریقی اجزای سیستم ممکن است دچار خطا شوند. هر یک از الگوریتم ها با برخی از مدل های خطا سازگار هستند و از این جهت شناخت دقیق مدل های خطا ضرورت دارد.

یک سیستم t -fault-tolerant نامیده می شود اگر با وجود حداکثر t جزء خطادار، بتواند هدف خود را محقق کند.

MTBF یا همان mean time between failures نشان دهنده زمان مورد انتظار تا بروز خطا می باشد.

خوش غیم

Process failure models (Benign)

Fail-stop

در این مدل، از یک لحظه به بعد پروسه ای که به درستی کار می کرده از کار می افتد. سایر پروسه ها می فهمند که این پروسه دچار اشکال شده است. این مدل یک مدل انتزاعی است و نحوه فهم دیگران از خطا می تواند کاملاً متنوع باشد.

Crash

در این مدل یک پروسه که درست کار می کرده در یک لحظه از کار می افتد. نودهای دیگر از آن مطلع نمی شوند.

Receive omission:

در این مدل یک پروسه درست، به نحوی دچار خطا می شود که فقط برخی از پیامهایی را که برای آن ارسال شده است را دریافت می کند.

Send omission:

پروسه فقط بعضی از پیام هایی را که باید ارسال کند، واقعاً ارسال می کند.

General omission:

پروسه دچار یکی یا هر دو خطای فوق می شود.

Process failure models (Byzantine)

Byzantine or malicious failure, with authentication

پروسه ممکن است هر رفتار دلخواهی را از خود نشان دهد. اما چنانچه ادعا کند که پیامی را از نود خاصی دریافت کرده است، این ادعا با مکانیزم authentication یا امضا قابل ارزیابی خواهد بود.

Byzantine or malicious failure

پروسه ممکن است هر رفتار دلخواهی نشان دهد و بر خلاف مدل ادعای دریافت پیام از یک نود سالم، قبل قابل ارزیابی نخواهد بود.

Communication failure models

Crash failure

یک لینک سالم از یک لحظه، هیچ پیامی را به مقصد نمی‌رساند.

Omission failure

لینک بعضی از پیام‌ها را به مقصد می‌رساند و بعضی را نمی‌رساند.

Byzantine failure

لینک ممکن است هر رفتار دلخواهی را بروز دهد. مثلاً پیامی ایجاد کند یا محتوای یک پیام را تغییر دهد.

علاوه بر خطاهای فوق‌خطای **timing** هم در مورد پروسه‌ها و هم لینک ارتباطی می‌تواند وجود داشته باشد که مختص سیستم‌های **synchronized** است. به این معنی که یک لینک پیام‌ها را کندتر یا سریع‌تر از آنچه باید، انتقال دهد.

مرجع: فصل ۵ کتاب Singhal

The coordinated attack problem

در مساله حمله هماهنگ چند ژنرال قرار است با نیروهایشان به یک هدف حمله کنند. در صورتی که همه به صورت هماهنگ حمله کنند، نتیجه موفقیت آمیز خواهد بود و در غیر اینصورت همگی نابود خواهند شد.

هر یک از ژنرال‌ها ذهنیت خود را از آمادگی نیروهایش دارد.

ژنرال‌ها در مکان‌های مختلفی مستقر شده‌اند. ارتباط بین ژنرال‌ها از طریق پیک‌هایی است که ممکن است اسیر یا کشته شوند. با توجه به اینکه راه ارتباطی بین آنها غیر قابل اطمینان است، ژنرال‌ها باید به طریقی هماهنگ با هم عمل کنند. یا همگی حمله کنند یا هیچ کدام. اولویت با حمله می‌باشد.

The coordinated attack problem

در صورتی که همه پیک ها قابل اطمینان بودند، کافی بود هر یک از ژنرال ها پیامی را به سایر ژنرال ها می فرستاد و از تصمیم آن ها سوال می کرد. سپس بعد از چند مرحله (قطر گراف ارتباطی)، همه ژنرال ها از همه تصمیمات با خبر می شدند. به عبارتی اگر همه آماده بودند، حمله آغاز می شد.

این مساله همان مساله distributed commit در دیتابیس های توزیع شده است.

در صورتی که امکان گم شدن پیک یا پیام وجود داشته باشد، این روش قابل اجرا نیست.

اثبات می شود که هیچ الگوریتمی وجود ندارد که همیشه بتواند این مساله را حل کند.

مدل مساله

n نود که از ۱ تا n شماره گذاری شده اند.

گراف ارتباطی بدون جهت می باشد.

هر نود از کل گراف مطلع است.

تصمیم هر نود ۰ یا ۱ است. ۱ به معنی حمله یا کامیت خواهد بود.

شرایط تصمیم گیری

□ Safety:

□ Agreement: no two processes decide on different values.

□ Validity:

اگر همه نودها با ۰ شروع کردند، نتیجه نهایی باید ۰ باشد.

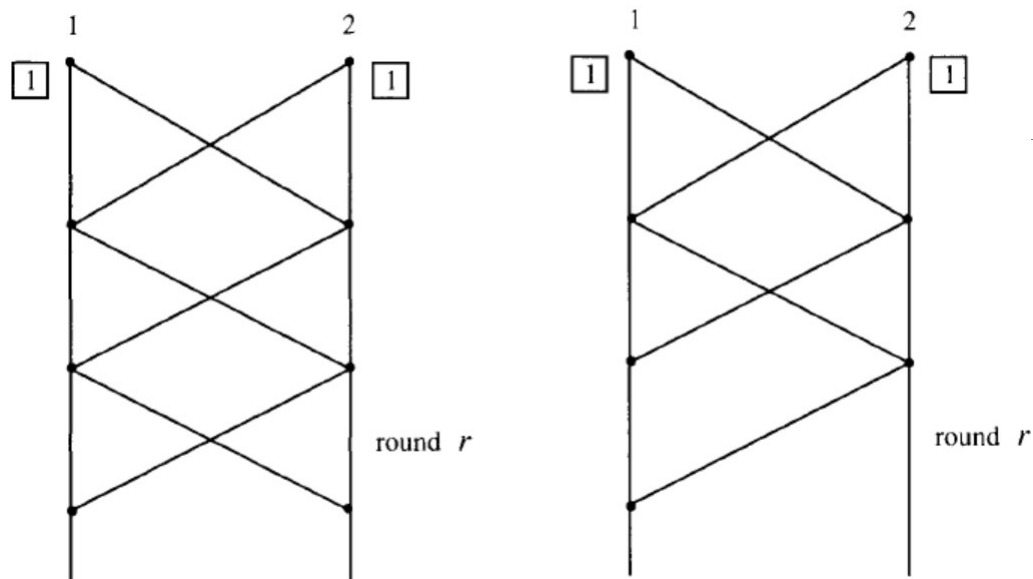
اگر همه با ۱ شروع کردند و هیچ پیامی گم نشد، نتیجه نهایی باید ۱ باشد.

□ Liveness:

□ Termination: All processes eventually decide.

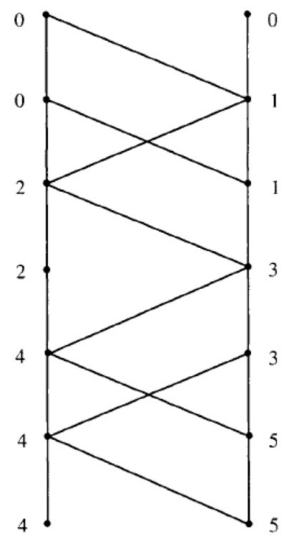
□ Fault Tolerance

No deterministic consensus protocol provides all three of safety, liveness, and fault tolerance in an asynchronous system.



Randomized Algorithm

A simplifying assumption is that network is complete, although a strongly-connected network with r greater than or equal to the diameter also works.



Randomized Algorithm

- First part: tracking information levels
 - Each process tracks its “information level,” initially 0. The state of a process consists of a vector of (input, information-level) pairs for all processes in the system. Initially this is (my-input, 0) for itself and $(\perp, -1)$ for everybody else.
 - Every process sends its entire state to every other process in every round.
 - Upon receiving a message m , process i stores any inputs carried in m and, for each process j , sets $\text{level}_i[j]$ to $\max(\text{level}_i[j], \text{level}_m[j])$. It then sets its own information level to $\min_j(\text{level}_i[j]) + 1$.
- Second part: deciding the output
 - Process 1 chooses a random key value uniformly in the range $[1, r]$.
 - This key is distributed along with $\text{level}_i[1]$, so that every process with $\text{level}_i[1] \geq 0$ knows the key.
 - A process decides 1 at round r if and only if it knows the key, its information level is greater than or equal to the key, and all inputs are 1.

states:

rounds $\in \mathbb{N}$, initially 0

decision $\in \{\text{unknown}, 0, 1\}$, initially *unknown*

key $\in [1, r] \cup \text{undefined}$, initially *undefined*

for every j , $1 \leq j \leq n$:

$\text{val}(j) \in \{0, 1, \text{undefined}\}$; initially $\text{val}(i)$ is i 's initial value and

$\text{val}(j) = \text{undefined}$ for all $j \neq i$

$\text{level}(j) \in [-1, r]$; initially $\text{level}(i) = 0$ and $\text{level}(j) = -1$ for all $j \neq i$

rand_i:

if $i = 1$ and $rounds = 0$ then $key := random$

msg_s:

— send (L, V, key) to all j , where L is the *level* vector and V is the *val* vector —

trans_i:

$rounds := rounds + 1$

let (L_j, V_j, k_j) be the message from j , for each j from which a message arrives

if, for some j , $k_j \neq undefined$ then $key := k_j$

for all $j \neq i$ do

 if, for some i' , $V_{i'}(j) \neq undefined$ then $val(j) := V_{i'}(j)$

 if, for some i' , $L_{i'}(j) > level(j)$ then $level(j) := \max_{i'} \{L_{i'}(j)\}$

$level(i) := 1 + \min \{level(j) : j \neq i\}$

if $rounds = r$ then

 if $key \neq undefined$ and $level(i) \geq key$ and $val(j) = 1$ for all j then

$decision := 1$

 else $decision := 0$

منبع:

فصل 5 از کتاب Lynch

تمرین:

الگوریتم حمله هماهنگ رندوم را شبیه سازی کنید. برای این منظور چند نود در نظر بگیرید که به صورت تصادفی تصمیم گیری اولیه انجام می دهند. سپس با اجرای الگوریتم و با فرض اینکه برخی از پیام ها به صورت رندوم دریافت نمی شوند، مساله حمله هماهنگ را شبیه سازی کنید. تحلیل کنید که شرایط سه گانه تا چه حد برآورده می شود.

همچنین تحلیل کنید که در چند درصد مواقع الگوریتم جواب متفاوتی می دهد.

سورس برنامه شبیه سازی به علاوه گزارش در قالب PDF را تحویل دهید. گزارش به تنهایی باید جامع باشد و نیازی به مراجعه به برنامه نداشته باشد.