gen_paxos

kuenishi

2009/7/3 3rd Erlang-meeting in Shirogane-takanawa, Tokyo

自己紹介

- ウエニシ コウタ(kuenishi@gmail.com)

- 趣味
  - 役に立つかどうか分かりもしないプログラムをつくること
  - ErlangやPython
  - See id:kuenishi, g:erlang:id:kuenishi, @kuenishi
    - http://github.com/kuenishi
    - http://bitbucket.org/kuenishi

- Recent Activities
  - Yet another TC-Erlang binding
    - http://bitbucket.org/kuenishi/yatce/
  - Mercurial l10n
    - 1.3 has been released on 7/1!
    - http://bitbucket.org/foozy/mercurial-translation-ja/

Outline

- Introduction
- Why PAXOS?
- What is PAXOS?
- How can we implement PAXOS?
    - Why Erlang?
- Where is PAXOS?
- When?
- Appendix

- Notice: Basically English, Sometimes Japanese

Introduction – what is gen_paxos?

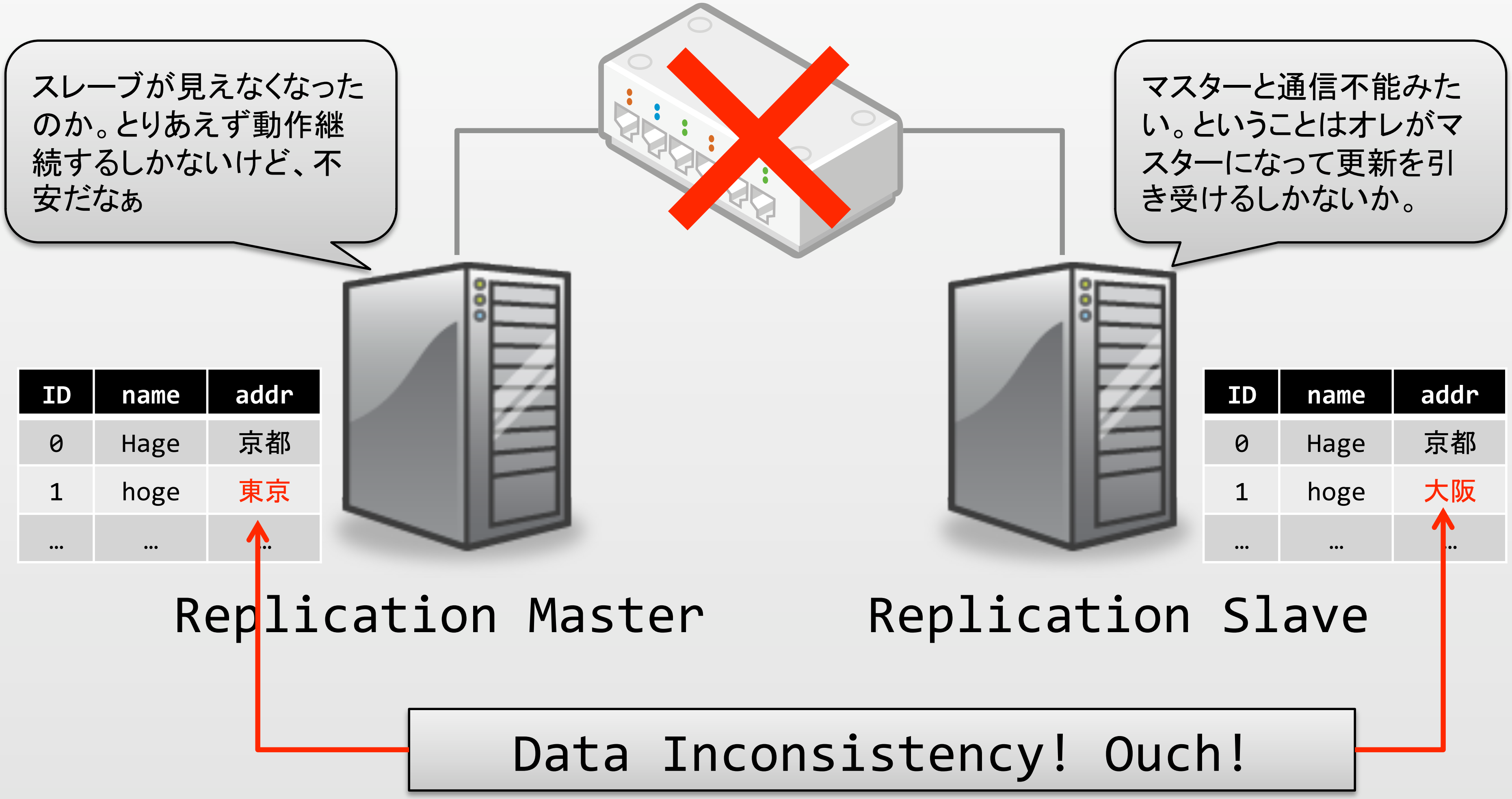
- GENeric PAXOS module written in Erlang
    - Reduction of the cost of re-invention of wheels.
    - At first, I had aimed to make it sth like ‘behaviour’
    - http://github.com/kuenishi/gen_paxos

- Related works (I think not enough)
    - libpaxos
    - gen_leader

    - Why not enough?

Why PAXOS?

- Split Brain
  - Typically, broken switches / wires
  - Both master and slave **updates independently**


スレーブが見えなくなったのか。とりあえず動作継続するしかないけど、不安だなぁ
マスターと通信不能みたい。ということはオレがマスターになって更新を引き受けるしかないか。
Replication Master
Replication Slave
Data Inconsistency! Ouch!

ID	name	addr
0	Hage	京都
1	hoge	東京
…	…	…

ID	name	addr
0	Hage	京都
1	hoge	大阪
…	…	…

Why PAXOS? – cont’d

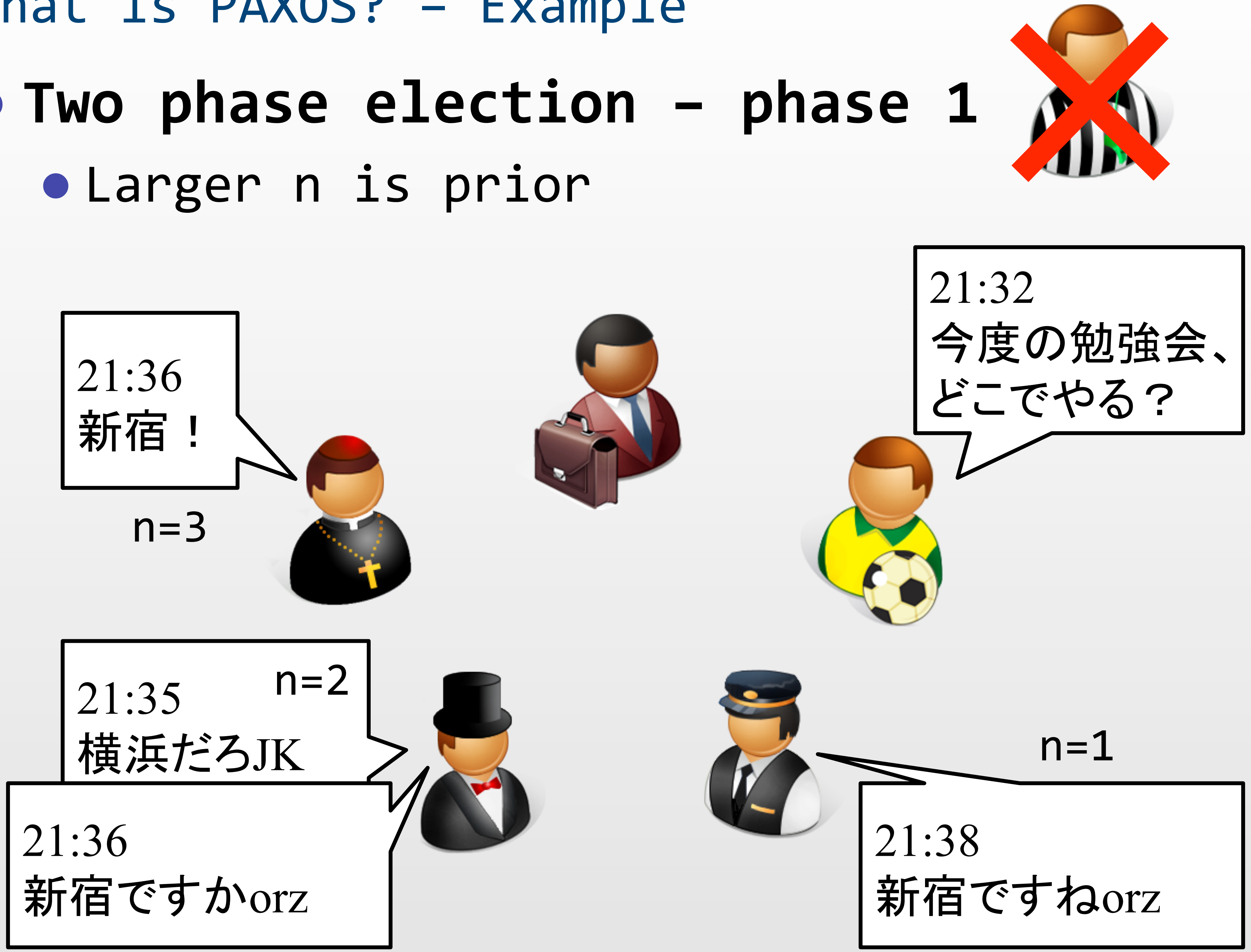
- Split Brain Problem
  - Separate actors can’t coordinate
    - Lacking coordination -> Multi-master
    - Multi-master -> Data Inconsistency
    - Inconsistency -> Tragedy
  - Typically, BigTable solely depends upon Chubby[2]
    - BigTable’s Master Election, so on.
- Byzantine Generals Problem
  - A faulty actor who lies destroys consistency
  - I don’t know current implementation of gen_paxos solves this problem – not in scope
  - See Byzantine PAXOS in Wikpedia[6]

What is PAXOS?

- **Distributed Consensus/Coordination Algorithm**
  - Masterless (no SPOF)
  - Robust to split brain environment such as network separation
  - Proved to converge within infinite time period[9]
  - Works iff majority of fixed group of actors is alive and communicable

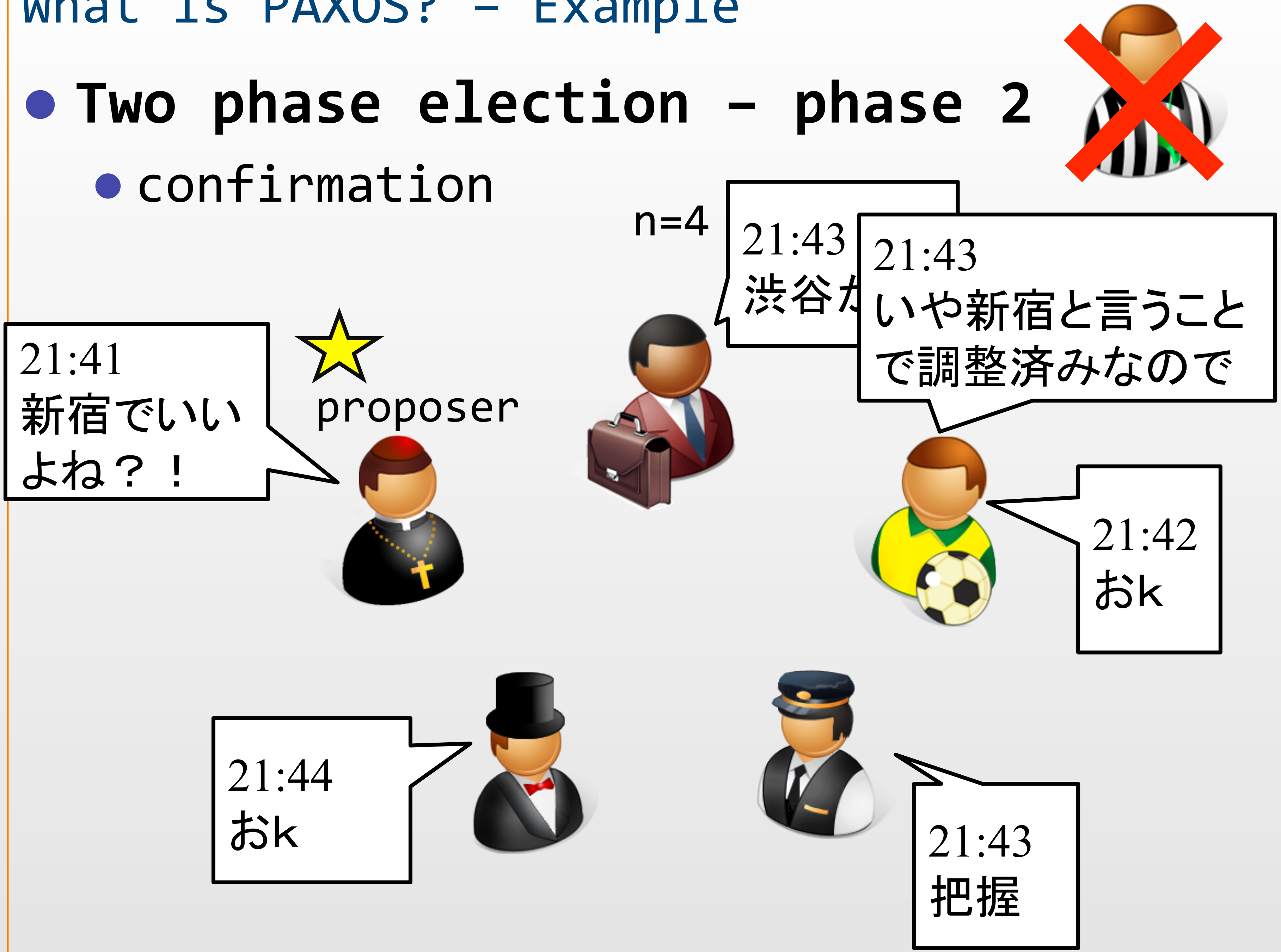
What is PAXOS? – Example

- **Two phase election – phase 1**
  - Larger n is prior



What is PAXOS? – Example

- **Two phase election – phase 2**
  - confirmation



What is PAXOS? – data model in gen_paxos

- Actors
  - n processors with processor ID
- Data
  - Instance ID (Subject)
  - Proposal ID (N)
  - Proposal itself (Value)
- N should be unique
- At first, V differs
  - later V converges

ID	1	2	3	4	5
n(1)	1	2	3	4	5
n(2)	6	7	8	9	10
n(3)	11	12	13	14	15
n(4)	16	17	18	19	20
n(…)	…	…	..	…	..

What is PAXOS? – details in gen_paxos

- Basic sequence of Each actor

- <u>Prepare Phase</u>
  - If Prepare comes, agree or disagree (be Acceptor)
  - If Prepare doesn’t come, send Prepare (be Proposer)
  - If majority of group agrees Prepare, send Propose
  - If timeout, increase n and restart Prepare phase
- <u>Propose Phase</u>
  - Acceptor – agree to Propose (ignoring lower n)
  - Proposer – decided if majority agrees
  - If timeout, increase n and restart Prepare phase
- <u>Commit Phase</u>
  - Send the committed Value again to notify

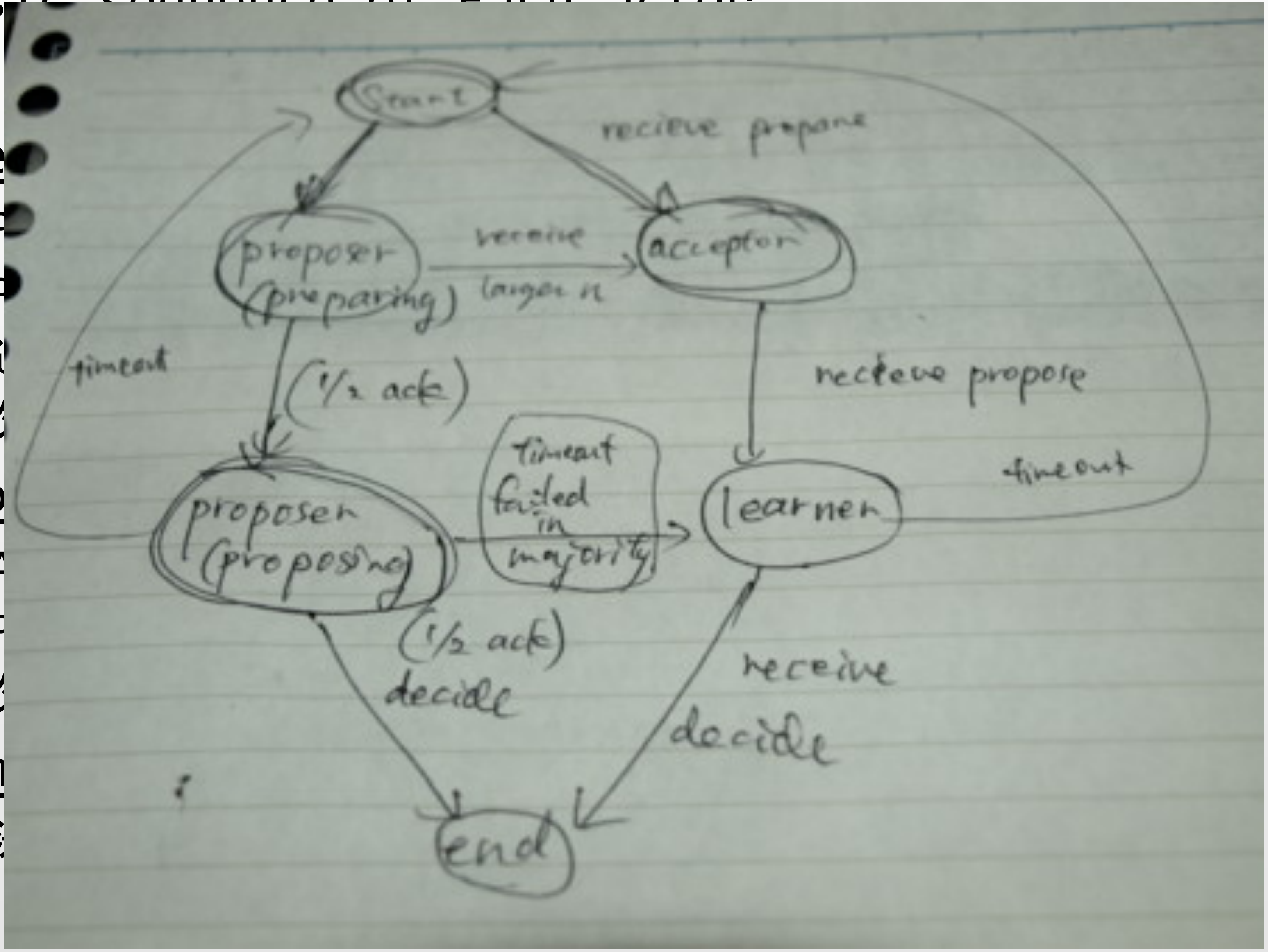
What is PAXOS? – details in gen_paxos

- Basic sequence of Each actor

- Prepare Phase
  - Prepareが来たら賛成したりしなかったり（Acceptor）
  - Prepareが来なかったらPrepareを送る（Proposer）
  - 過半数からPrepareの賛成が来たらPropose
  - タイムアウトしたらnを増やして戻る
- Propose Phase
  - Acceptor - Proposeが来たら賛成（nの値によっては無視）
  - Proposer - 過半数から賛成が来たら決定
  - タイムアウトしたらnを増やしてPrepare Phaseに戻る
- Commit Phase
  - 決まった値をもう一度送る

What is PAXOS? - details in gen_paxos

- Basic sequence of Each actor
- Pre
  - P
  - P
  - 辶
  - 分
- Pro
  - A
  - P
  - 分
- Com
  - 氵

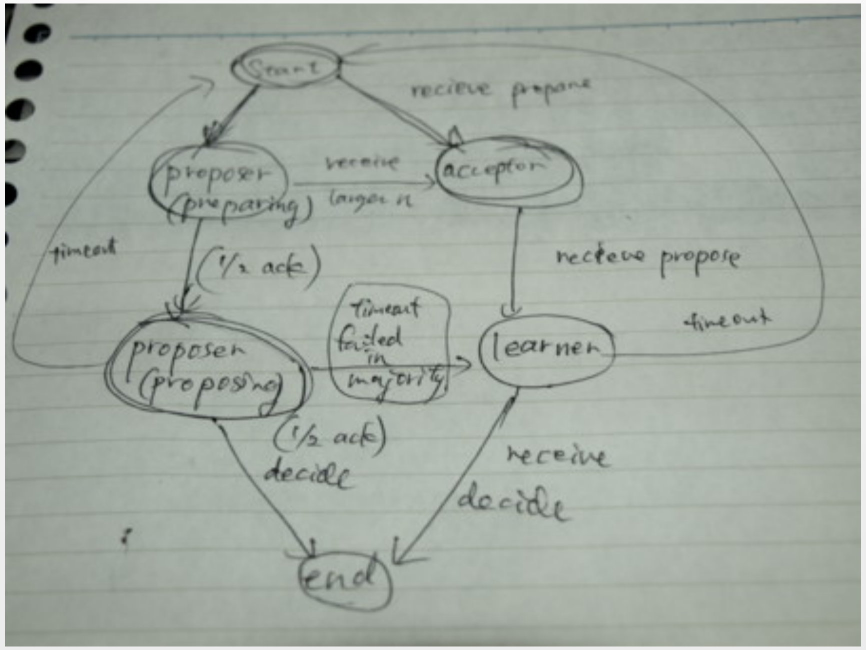


How can we implement PAXOS?

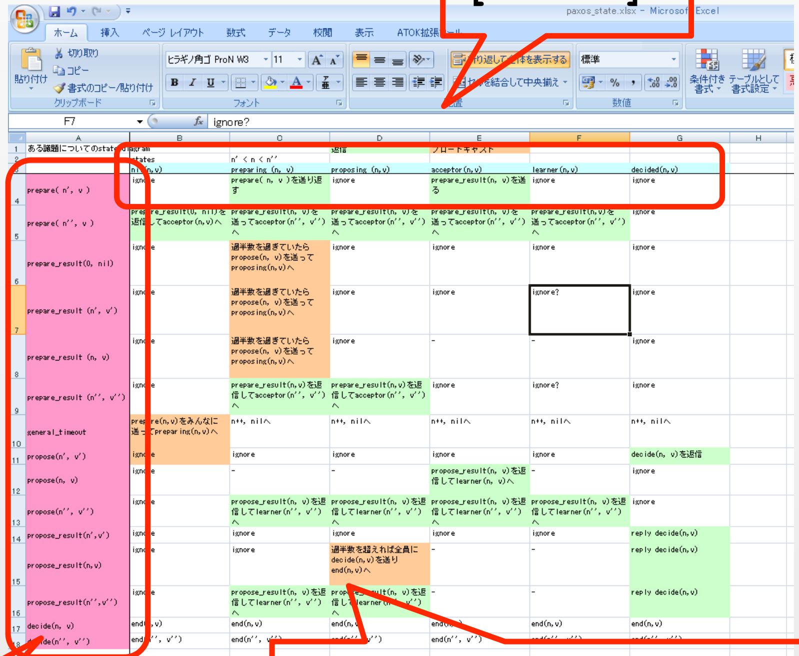
1. **Make a State Diagram**
2. **Write an FSM code**

How can we implement PAXOS? (1/2)

1. Make a State Diagram



[State]



F7 ignore?

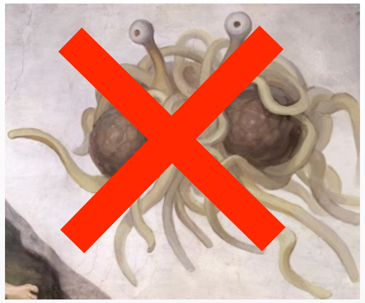
	A	B	C	D	E	F	G
1	ある議題についてのstate diagram			返信	ブロードキャスト		
2		states	n' < n < n''				
3		nil(n,v)	preparing (n, v)	proposing (n,v)	acceptor(n,v)	learner(n,v)	decided(n,v)
4	prepare(n', v)	ignore	prepare(n, v)を送り返す	ignore	prepare_result(n, v)を送る	ignore	ignore
5	prepare(n'', v)	prepare_result(0, nil)を返信してacceptor(n,v)へ	prepare_result(n, v)を送ってacceptor(n'', v'')へ	prepare_result(n, v)を送ってacceptor(n'', v'')へ	prepare_result(n, v)を送ってacceptor(n'', v'')へ	prepare_result(n,v)を送ってacceptor(n'', v'')へ	ignore
6	prepare_result(0, nil)	ignore	過半数を過ぎていたらpropose(n, v)を送ってproposing(n,v)へ	ignore	ignore	ignore	ignore
7	prepare_result (n', v')	ignore	過半数を過ぎていたらpropose(n, v)を送ってproposing(n,v)へ	ignore	ignore	ignore?	ignore
8	prepare_result (n, v)	ignore	過半数を過ぎていたらpropose(n, v)を送ってproposing(n,v)へ	ignore	-	-	ignore
9	prepare_result (n'', v'')	ignore	prepare_result(n,v)を返信してacceptor(n'', v'')へ	prepare_result(n,v)を返信してacceptor(n'', v'')へ	ignore	ignore?	ignore
10	general_timeout	prepare(n,v)をみんなに送ってpreparing(n,v)へ	n++, nilへ	n++, nilへ	n++, nilへ	n++, nilへ	n++, nilへ
11	propose(n', v')	ignore	ignore	ignore	ignore	ignore	decide(n, v)を返信
12	propose(n, v)	ignore	-	-	propose_result(n, v)を返信してlearner(n, v)へ	-	ignore
13	propose(n'', v'')	ignore	propose_result(n, v)を返信してlearner(n'', v'')へ	propose_result(n, v)を返信してlearner(n'', v'')へ	propose_result(n, v)を返信してlearner(n'', v'')へ	propose_result(n, v)を返信してlearner(n'', v'')へ	ignore
14	propose_result(n',v')	ignore	ignore	ignore	ignore	ignore	reply decide(n,v)
15	propose_result(n,v)	ignore	ignore	過半数を超えれば全員にdecide(n,v)を送りend(n,v)へ	-	-	reply decide(n,v)
16	propose_result(n'',v'')	ignore	propose_result(n, v)を返信してlearner(n'', v'')へ	propose_result(n, v)を返信してlearner(n'', v'')へ	-	-	reply decide(n,v)
17	decide(n, v)	end(n,v)	end(n,v)	end(n,v)	end(n,v)	end(n,v)	end(n,v)
18	decide(n'', v'')	end(n'', v'')	end(n'', v'')	end(n'', v'')	end(n'', v'')	end(n'', v'')	end(n'', v'')

{Action, NextState}

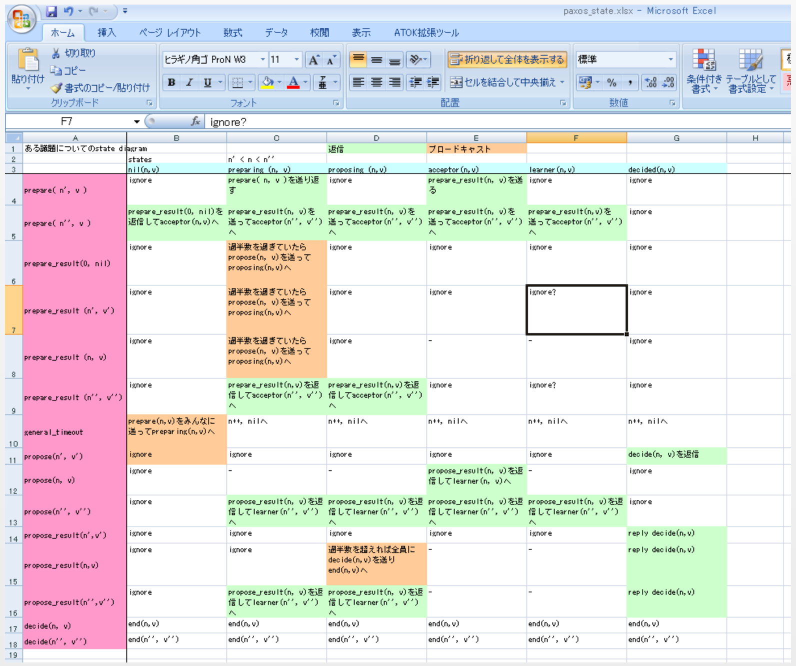
[Event]

Define:
- {State, Event} -> {NextState, Action}

How can we implement PAXOS? (2/2)



2. Write an FSM code


paxos_state.xlsx - Microsoft Excel
ホーム 挿入 ページ レイアウト 数式 データ 校閲 表示 ATOK拡張ツール
F7
ignore?

	A	B	C	D	E	F	G
1	ある議題についてのstate diagram			返信	ブロードキャスト		
2		states	n' < n < n''				
3		nil(n,v)	preparing (n, v)	proposing (n,v)	acceptor(n,v)	learner(n,v)	decided(n,v)
4	prepare(n', v)	ignore	prepare(n, v)を送り返す	ignore	prepare_result(n, v)を送る	ignore	ignore
5	prepare(n'', v)	prepare_result(0, nil)を返信してacceptor(n,v)へ	prepare_result(n, v)を送ってacceptor(n'', v'')へ	prepare_result(n, v)を送ってacceptor(n'', v'')へ	prepare_result(n, v)を送ってacceptor(n'', v'')へ	prepare_result(n,v)を送ってacceptor(n'', v'')へ	ignore
6	prepare_result(0, nil)	ignore	過半数を過ぎていたらpropose(n, v)を送ってproposing(n,v)へ	ignore	ignore	ignore	ignore
7	prepare_result (n', v')	ignore	過半数を過ぎていたらpropose(n, v)を送ってproposing(n,v)へ	ignore	ignore	ignore?	ignore
8	prepare_result (n, v)	ignore	過半数を過ぎていたらpropose(n, v)を送ってproposing(n,v)へ	ignore	-	-	ignore
9	prepare_result (n'', v'')	ignore	prepare_result(n,v)を返信してacceptor(n'', v'')へ	prepare_result(n,v)を返信してacceptor(n'', v'')へ	ignore	ignore?	ignore
10	general_timeout	prepare(n,v)をみんなに送ってpreparing(n,v)へ	n++, nilへ	n++, nilへ	n++, nilへ	n++, nilへ	n++, nilへ
11	propose(n', v')	ignore	ignore	ignore	ignore	ignore	decide(n, v)を返信
12	propose(n, v)	ignore	-	-	propose_result(n, v)を返信してlearner(n, v)へ	-	ignore
13	propose(n'', v'')	ignore	propose_result(n, v)を返信してlearner(n'', v'')へ	propose_result(n, v)を返信してlearner(n'', v'')へ	propose_result(n, v)を返信してlearner(n'', v'')へ	propose_result(n, v)を返信してlearner(n'', v'')へ	ignore
14	propose_result(n',v')	ignore	ignore	ignore	ignore	ignore	reply decide(n,v)
15	propose_result(n,v)	ignore	ignore	過半数を超えれば全員にdecide(n,v)を送りend(n,v)へ	-	-	reply decide(n,v)
16	propose_result(n'',v'')	ignore	propose_result(n, v)を返信してlearner(n'', v'')へ	propose_result(n, v)を返信してlearner(n'', v'')へ	-	-	reply decide(n,v)
17	decide(n, v)	end(n,v)	end(n,v)	end(n,v)	end(n,v)	end(n,v)	end(n,v)
18	decide(n'', v'')	end(n'', v'')	end(n'', v'')	end(n'', v'')	end(n'', v'')	end(n'', v'')	end(n'', v'')
19							

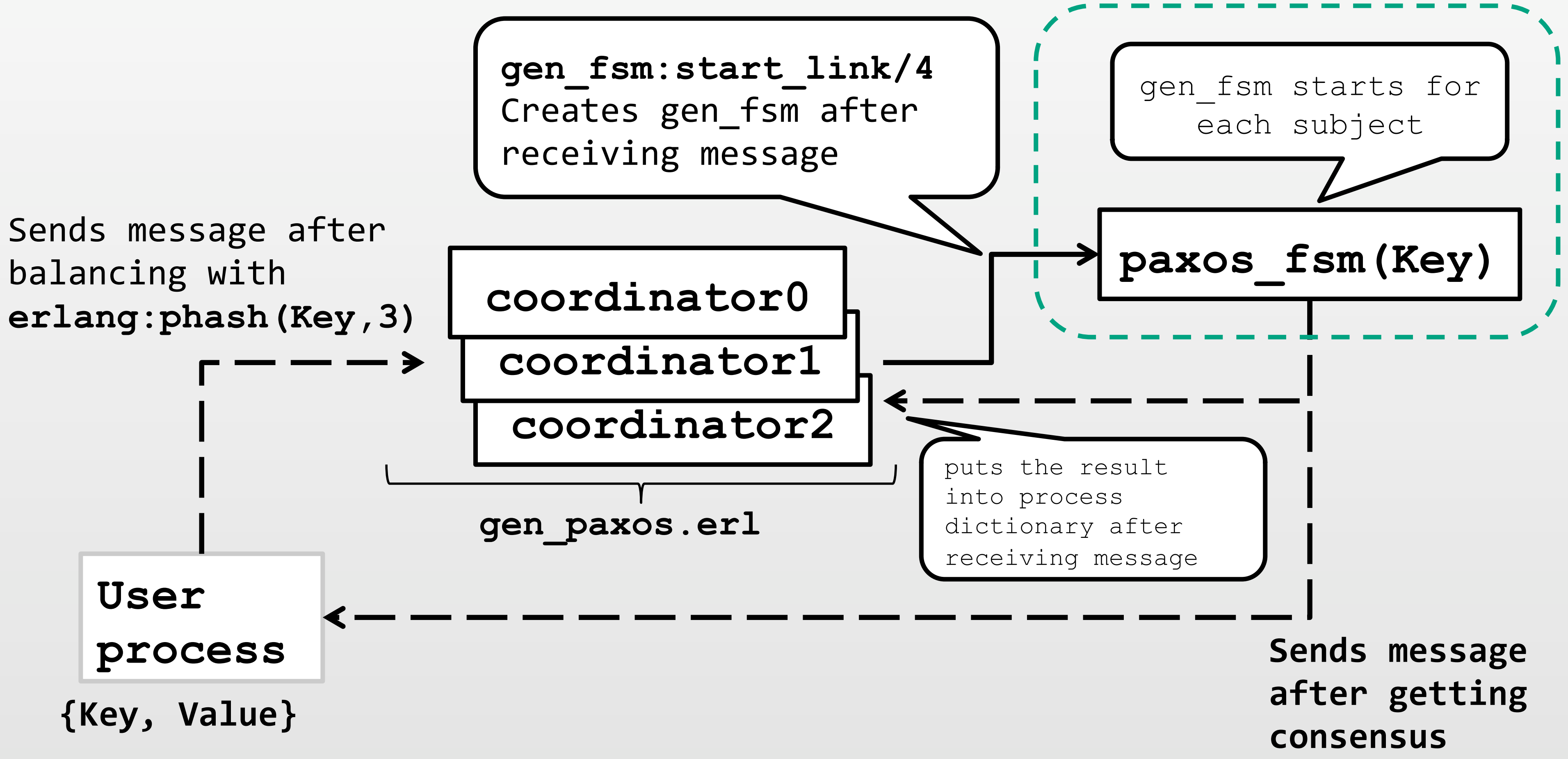
Define a set of Module:State/2

```
Module:State(Event, Data)
    -> {next_state,
        NextState, NextData}
```

(See paxos_fsm.erl for details)

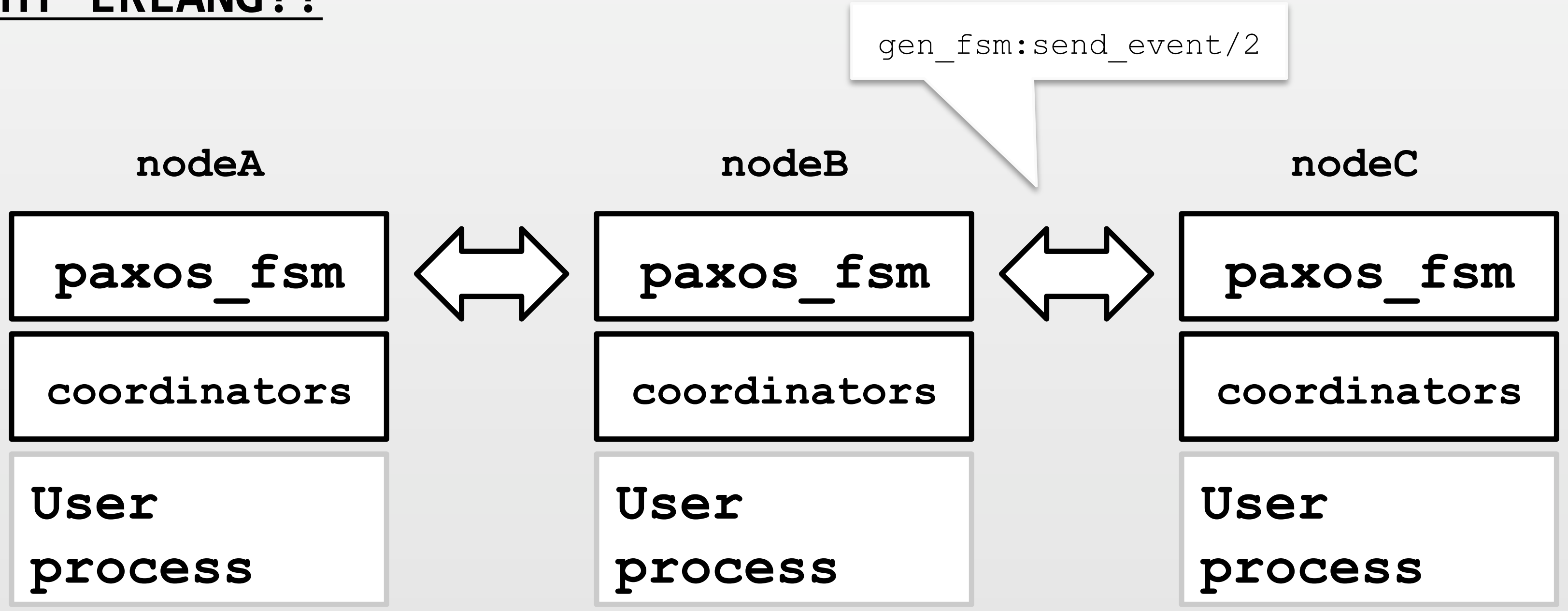
Process Architecture of a Node (gen_paxos.erl)

- A-little-bit load balancing
- Not pursuing consensus performance
- Thinkin'bout Chubby[5] (no dynamic node addition/removal)

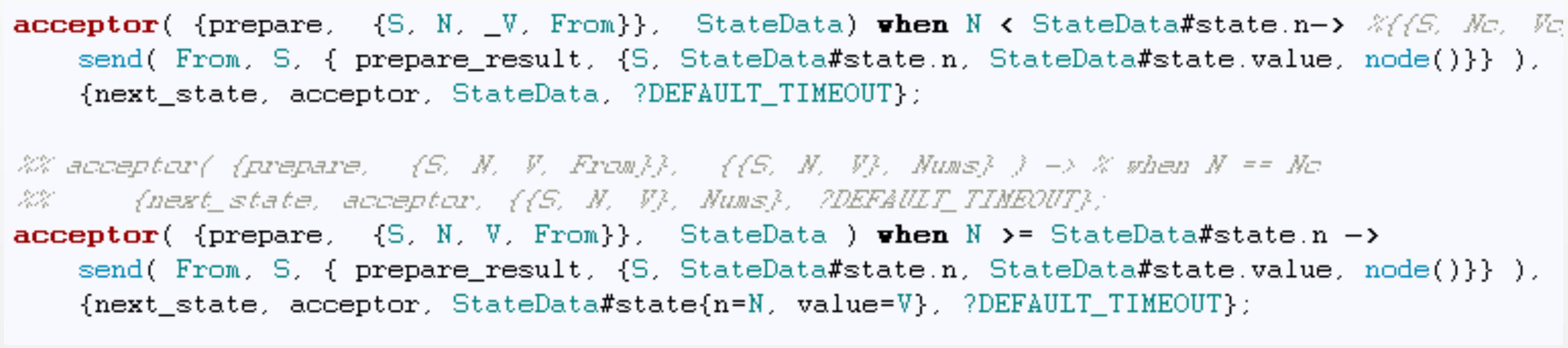

gen_fsm:start_link/4
Creates gen_fsm after receiving message
gen_fsm starts for each subject
Sends message after balancing with erlang:phash(Key,3)
coordinator0
coordinator1
coordinator2
paxos_fsm(Key)
gen_paxos.erl
puts the result into process dictionary after receiving message
User process
{Key, Value}
Sends message after getting consensus

Messaging between nodes (paxos_fsm.erl)

- FSMs talk each other by sending events with globalname()
  - `globalname() = {global, GlobalName}`
  - GlobalName is for first argument of gen_fsm:start_link/4
- gen_fsm:send_event/2 sends Events to **remote nodes**
  - `gen_fsm:send_event({global, GlobalName}, Event).` – that's all.
- Anyway, no need to design application protocol, object serialization, nor GoF 'State, Observer' pattern! **That's WHY ERLANG!!**



Sample code – paxos_fsm.erl



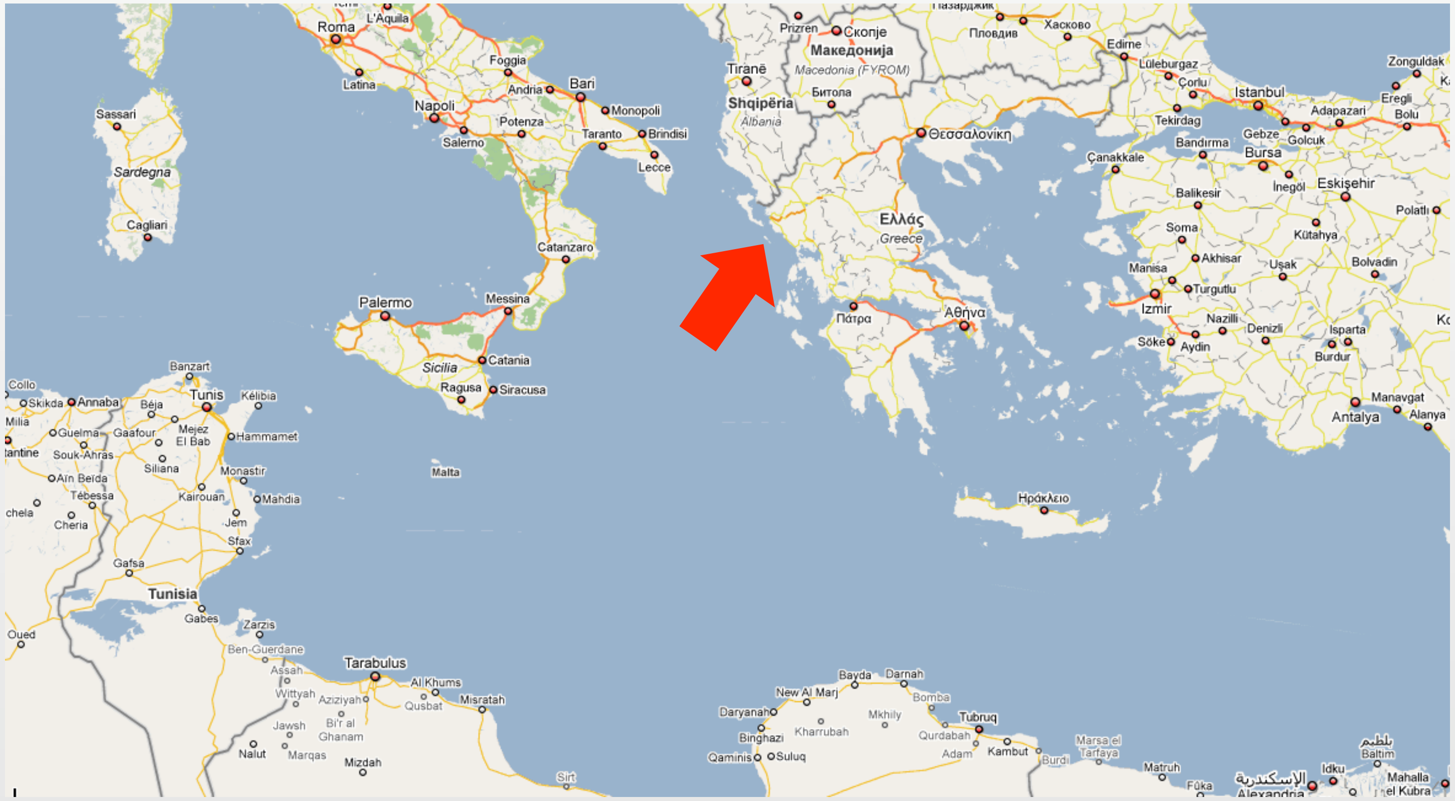
```
acceptor( {prepare,  {S, N, _V, From}},  StateData) when N < StateData#state.n-> %{{S, Nc, Vc
    send( From, S, { prepare_result, {S, StateData#state.n, StateData#state.value, node()}} ),
    {next_state, acceptor, StateData, ?DEFAULT_TIMEOUT};

%% acceptor( {prepare,  {S, N, V, From}},  {{S, N, V}, Nums} ) -> % when N == Nc
%%      {next_state, acceptor, {{S, N, V}, Nums}, ?DEFAULT_TIMEOUT};
acceptor( {prepare,  {S, N, V, From}},  StateData ) when N >= StateData#state.n ->
    send( From, S, { prepare_result, {S, StateData#state.n, StateData#state.value, node()}} ),
    {next_state, acceptor, StateData#state{n=N, value=V}, ?DEFAULT_TIMEOUT};
```

読み方
- acceptorのときに{prepare, {S,N,V,From}}(自分より大きなN)なイベントを受け取ったら、
- prepare_resultを返信して
- NとVを更新してacceptor状態へ移る。

Where is PAXOS?

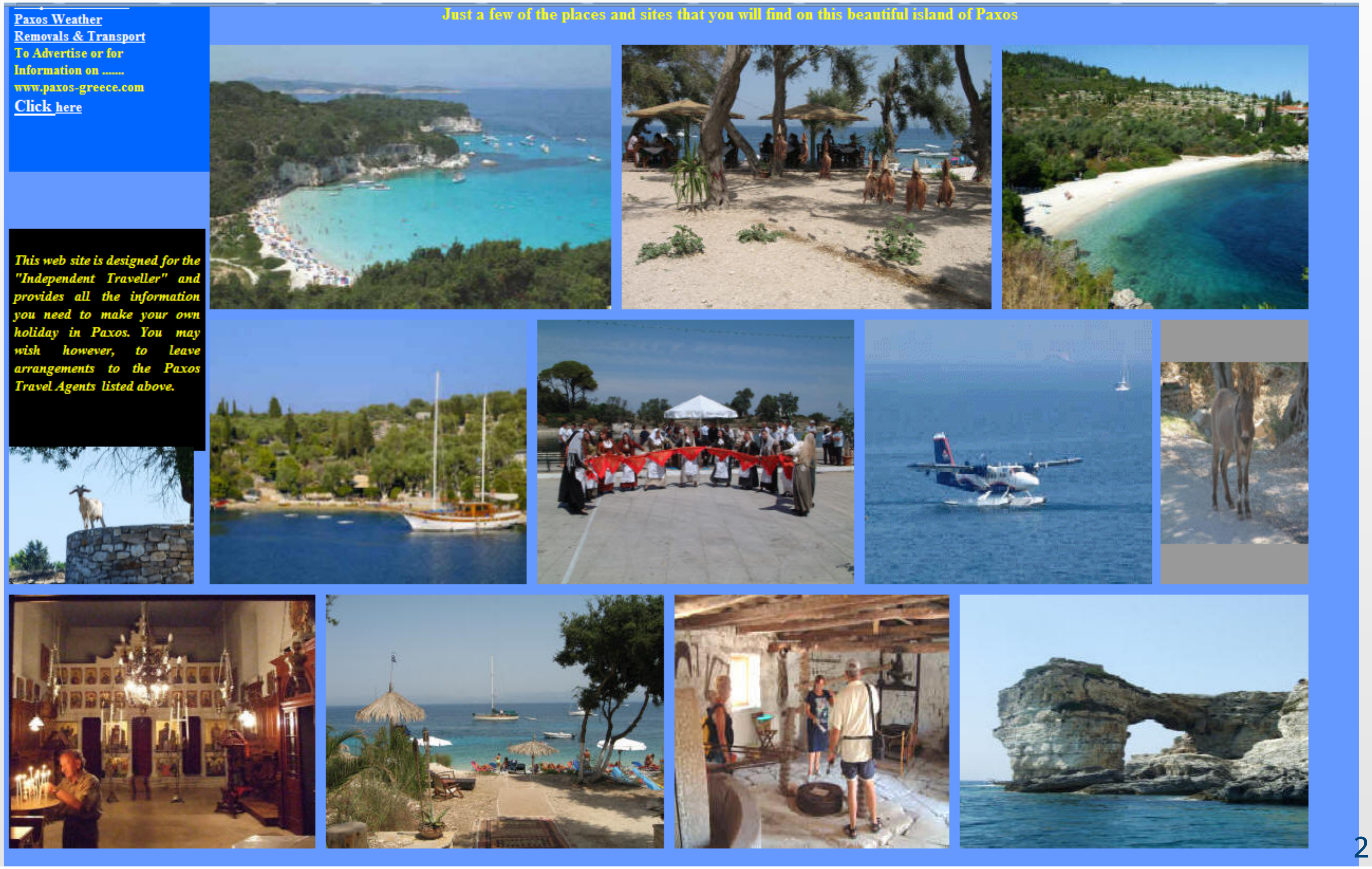
Where is PAXOS?

- Paxon Isles in Aegean Sea[1]
  - ‘the part-time parliament’[3]



Where is PAXOS? cont'd

- Looks so nice place to visit!



Paxos Weather
Removals & Transport
To Advertise or for Information on ....
www.paxos-greece.com
Click here

Just a few of the places and sites that you will find on this beautiful island of Paxos

This web site is designed for the "Independent Traveller" and provides all the information you need to make your own holiday in Paxos. You may wish however, to leave arrangements to the Paxos Travel Agents listed above.

When?? - References

- **[1]PAXOS island**
  - **http://www.paxos-greece.com/**
- **[2]Google, The Chubby Paper, 2006.**
- **[3]Leslie Lamport, \`The Part-Time Parliament'**, ACM Trans. Comput. Syst. Volume 16 , Issue 2 (May 1998) , pp. 133 – 169.
- **[4]Leslie Lamport, \`Paxos Made Simple'**, *SIGACT News*, Vol. 32, No. 4. (December 2001), pp. 51-58.
- **[5]T Chandra et al., \`Paxos made live: an engineering perspective'**, *PODC '07*, ACM Press, 2007, pp. 398-407.
- **[6]Wikipedia, \`Paxos_algorithm',**
  - **http://en.wikipedia.org/wiki/Paxos_algorithm**
  - 1990年くらいにはもうネタはできていたらしい
- **[7]Lamport's patent （さすがM$…）**
  - **http://www.j-tokkyo.com/2005/G06F/JP2005-196763.shtml**
- **[8]David Mazieres, \`Paxos Made Practical'**, tech report, 2007.
  - **http://net.pku.edu.cn/~course/cs501/2008/reading.html**
- **[9]Dwork, Cynthia et al., \`Consensus in the presence of partial synchrony'**, J. ACM 35-2, pp. 288–323, 1988.

- **Other implementations**
  - single-phase majority-voting is used in distributed memory, distributed CPU; rather hardware.
  - libpaxos

- Any Questions?

appendix

Known Bugs
FAQ
State Diagram

Appendix – Known Bug(s)

- learnerの意味を勘違いした実装になっています
  - gen_paxosでは、acceptした後になる状態のこと → 間違い
  - 本当は、議決には参加しないで結果に従うノードのこと

- http://cooldaemon.tumblr.com/post/130422117/gen-paxos
  - 幾つかのノードを起動し、幾つかの議題を出してみた。
    その後、過半数のノードを除去し、議題を出すと、ひたすら合意を取り続ける。
    この状態で、除去したノードと同じ名前のノードを復活させ、net_adm_ping/1 で繋ぐ。
    予想では、ここで合意が取れると思ったが、駄目だった。

＿|￣|○

Appendix – FAQ

- 議題ごとに gen_fsm を使い捨てているので
  どっかにプロセスをストックしておいて
  再利用できないもんかなぁ？（優先度低）
  - A. Erlangの思想から言って(プロセスは低コスト)、使い捨てすべきだと思います。今は作りっぱなしなので、どこかで破棄するタイミングがあるとよいかも。

- ノードの追加・除去に対応できるよう、ノードの一覧は外だしにした方が良いかも。
  あとは、合意を取る最大回数も欲しい。
  - A. Chubbyはノードの動的追加・除去を前提としてないので外だしにはしていません。
  - A. nに上限を設けておいて、それをみんなが超えたら失敗、という作りはアリだと思います。
  - 結局、私に使い方（ユースケース）があまり見えていないままとりあえず書いたのが問題で、Chubby以外の便利な使い方があれば教えてください。

- 合意の間隔と回数は、起動時のグローバル値と、合意単位で設定するローカル値に分割すると使いやすいかも？
  - A. timeout時間はマクロ値（?DEFAULT_TIMEOUT）。早く決めたいときには短めに、確実に決めたいときは長めに設定できるように、start_linkの引数とか#stateのメンバにするのはよいかも。

Appendix – State Diagram

ある議題についてのstate diagram

返信　ブロードキャスト

	states	n' < n < n"				
	nil(n,v)	preparing (n, v)	proposing (n,v)	acceptor(n,v)	learner(n,v)	decided(n,v)
prepare(n', v)	ignore	prepare(n, v)を送り返す	ignore	prepare_result(n, v)を送る	ignore	ignore
prepare(n", v)	prepare_result(0, nil)を返信してacceptor(n,v)へ	prepare_result(n, v)を送ってacceptor(n", v")へ	prepare_result(n, v)を送ってacceptor(n", v")へ	prepare_result(n, v)を送ってacceptor(n", v")へ	prepare_result(n,v)を送ってacceptor(n", v")へ	ignore
prepare_result(0, nil)	ignore	過半数を過ぎていたらpropose(n, v)を送ってproposing(n,v)へ	ignore	ignore	ignore	ignore
prepare_result (n', v')	ignore	過半数を過ぎていたらpropose(n, v)を送ってproposing(n,v)へ	ignore	ignore	ignore?	ignore
prepare_result (n, v)	ignore	過半数を過ぎていたらpropose(n, v)を送ってproposing(n,v)へ	ignore	-	-	ignore
prepare_result (n", v")	ignore	prepare_result(n,v)を返信してacceptor(n", v")へ	prepare_result(n,v)を返信してacceptor(n", v")へ	ignore	ignore?	ignore
general_timeout	prepare(n,v)をみんなに送ってpreparing(n,v)へ	n++, nilへ	n++, nilへ	n++, nilへ	n++, nilへ	n++, nilへ
propose(n', v')	ignore	ignore	ignore	ignore	ignore	decide(n, v)を返信
propose(n, v)	ignore	-	-	propose_result(n, v)を返信してlearner(n, v)へ	-	ignore
propose(n", v")	ignore	propose_result(n, v)を返信してlearner(n", v")へ	propose_result(n, v)を返信してlearner(n", v")へ	propose_result(n, v)を返信してlearner(n", v")へ	propose_result(n, v)を返信してlearner(n", v")へ	ignore
propose_result(n',v')	ignore	ignore	ignore	ignore	ignore	reply decide(n,v)
propose_result(n,v)	ignore	ignore	過半数を超えれば全員にdecide(n,v)を送りend(n,v)へ	-	-	reply decide(n,v)
propose_result(n",v")	ignore	propose_result(n, v)を返信してlearner(n", v")へ	propose_result(n, v)を返信してlearner(n", v")へ	-	-	reply decide(n,v)
decide(n, v)	end(n,v)	end(n,v)	end(n,v)	end(n,v)	end(n,v)	end(n,v)
decide(n", v")	end(n", v")	end(n", v")	end(n", v")	end(n", v")	end(n", v")	end(n", v")

- Thanks!