```
{
    "_id": "dch",
    "_rev": "3-814a704a7e0c8d77c4fcbdde8b7bb744",
    "founder": [
        "http://jsonified.com/",
        "http://thecouchfirm.com/"
    ],
    "from": "New Zealand",
    "mails": [
        "dch@apache.org",
        "dch@jsonified.com"
    ],
    "name": "Dave Cottlehuber",
    "passions": [
        "Telemark Skiing",
        "CouchDB Fanatic"
    ],
    "twitter": "@dch__"
}
```

Ground Computing

- Tomorrow's problems will not be solved by "The Cloud"
- Building zero latency services
- In a time of unreliable networks
- With intermittent or event no connectivity
- On mobile & embedded devices
- Not just cloud & hosted services

Apache CouchDB

“Time to Relax”

- Native JSON-based document store
- Exposed over HTTP(S) API
- Built-in sync and doc conflict management
- MVCC-based snapshots of DB used for replication & indexes
- Indexes are implemented as map/reduce views over MVCC snapshot again
- - continuous updates streamable between nodes & clients
- - built in Erlang/OTP + JavaScript, some C sprinkled in (ICU, Snappy compression)
- - alternative implementations focus on clustering, distribution, privacy, portability ...

Why Port?

- P2P layer for CouchDB looks like a good fit
- Opens up Replication & Storage opportunities
- Use CouchDB as data store for PPSPP apps
- Fun!

- source: git://github.com/jsonified/swirl
- UDP translation layer **
- Native erlang layer *
- Dictionaries for options **
- Binary types for heavy data lifting
- Wrapped in scalable application layer *
- CouchDB as data store
- Hopefully it will scale & distribute well

Why PPSPP for CouchDB?

- P2P is part of CouchDB's community and culture
- ***P2P approach at both database & doc level offers a different set of tradeoffs than we can do today***
- CouchDB's database and view snapshots are similar to PPSPP root hash
- CouchDB's streaming _changes feed is like PPSPP live streaming
- as a possible secure alternative replication engine and db storage

- HTTP has inefficiencies
  - roundtrip overhead
  - TCP itself on mobile and LFN links
  - binary data requires annoying workarounds
  - hub architecture
  - security as an add-on
- Bit Torrent isn't right fit
  - always-on is not the future
  - no update support
  - managing trusted nodes & updates not straightforwards

Why Erlang/OTP for PPSPP?

- Dynamically typed functional language from Ericsson
- Concurrency without mutexes, threads, locks, or headaches
- Underlying BEAM VM is designed for ***scalable low-latency distributed systems***
- Ideal for low-level protocols both within a trusted network & without
- OTP libraries provide a superb robust & well tested platform
- Property-based testing & live tracing/debugging across VMs
- Just for kicks of implementing something from scratch :D

In Practice

```
(swerl@akai)7> swerl_app:start().
swerl: PPSPP release d-06
peer: listening on udp 7777
swerl: peer on 7777 registered as swerl_7777
<0.954.0>
peer: recv udp from 127.0.0.1:51722
dgram: recv on channel 0x00000000
parser: valid message type  handshake
dgram: parse ok on channel 0x00000000
peer: recv udp from 127.0.0.1:53790
dgram: recv on channel 0x00000000
parser: valid message type  handshake
dgram: parse ok on channel 0x00000000
peer: recv udp from 127.0.0.1:63326
...
```

And runs happily on multiple instances:

```
(swerl@akai)12> swerl_app:start(swerl, 7779).
swerl: PPSPP release d-06
peer: listening on udp 7779
swerl: peer on 7779 registered as swerl_7779
<0.16434.0>
(swerl@akai)13> swerl_app:start(swerl, 7778).
swerl: PPSPP release d-06
peer: listening on udp 7778
swerl: peer on 7778 registered as swerl_7778
<0.16444.0>
(swerl@akai)14>
```

Typical Recursive Parser

```
unpack(Transport, <<Channel:?PPSPP_CHANNEL_SIZE, Maybe_Messages/binary>> ) ->
    ?DEBUG_DGRAM_RECV(Channel),
    {ok, Parsed_Messages} = ppspp_message:unpack(Maybe_Messages),
    ?DEBUG_DGRAM_PARSE_OK(Channel),
    Datagram = orddict:store(messages, Parsed_Messages,
        orddict:store(channel, Channel, Transport)),
    {ok, Datagram}.

%% ... in another module far, far away...

ppspp_message:unpack(Maybe_Messages) when is_binary(Maybe_Messages) ->
    unpack(Maybe_Messages, []).

%% if the binary is empty, all messages were parsed successfully
unpack( <<>>, Parsed_Messages) ->
    {ok, lists:reverse(Parsed_Messages)};
%% otherwise try to unpack another valid message, peeling off and parsing
%% recursively the remainder, accumulating valid (parsed) messages.
%% A failure anywhere in a message ultimately causes the entire datagram
%% to be rejected.
unpack(<<Maybe_Message_Type:?PPSPP_MESSAGE_SIZE, Rest/binary>>, Parsed_Messages) ->
    {ok, Type} = validate_message_type(Maybe_Message_Type),
    [{Type, Parsed_Message}, Maybe_More_Messages] = parse(Type, Rest),
    unpack(Maybe_More_Messages, [Parsed_Message | Parsed_Messages]);
unpack(_Maybe_Messages, _Rest) -> {error, ppspp_invalid_message}.
```

PPSPP Options round trip

```
handshake_options(packed) -> Cat_hash = chairman_miaow_hash(),
<<  ?PPSPP_VERSION,                       1, %% ppspp version
    ?PPSPP_MINIMUM_VERSION,               1, %% ppspp max version
    ?PPSPP_SWARM_ID_LENGTH,      20:?WORD, %% swarm id length (160 bits)
    Cat_hash/binary                      , %% the merkle hash requested
    ?PPSPP_INTEGRITY_CHECK_METHOD,        1, %% optional integrity check method
    ?PPSPP_MERKLE_HASH_FUNCTION,          0, %% merkle please
    ?PPSPP_CHUNK_ADDRESSING_METHOD,       2, %% chunk addressing method 2
    ?PPSPP_END_OPTION >>; %% end options
handshake_options(unpacked) ->
    [{ppspp_chunking_method,ppspp_32bit_chunks},
        {ppspp_content_integrity_check_method,ppspp_merkle_hash_tree},
        {ppspp_merkle_hash_function,ppspp_sha1},
        {ppspp_minimum_version,1},
        {ppspp_swarm_id, chairman_miaow_hash()},
        {ppspp_version,1}].

dgram0() ->
    Msg0 = msg0(),
    << ?PPSPP_UNKNOWN_CHANNEL, Msg0/binary >>.

msg0() -> Type = ?HANDSHAKE,
    Channel = random_channel(),
    Options = handshake_options(packed),
    << Type/binary, %% handshake
    Channel/binary,    %% initiator peer channel
    Options/binary >>.
```

Credits

- Chairman Miaow http://obeythekitty.blogspot.de/
- Where's the balloon? http://cheezburger.com/6050984448
- Yoda cat http://dodger.uselessopinions.com/stuff/images/cat_yoda.jpg
-