Large Partially-connected Erlang Clusters

Motiejus Jakštys @mo_kelione
Sr. backend developer

spilgames

Erlang User Conference 2014
Stockholm

Background and Agenda

Big service-oriented backend infrastructure.

- Infrastructure which provides API for our gaming portals
- A couple of dozen servers running about 50 services each

Background and Agenda

Big service-oriented backend infrastructure.

- Infrastructure which provides API for our gaming portals
- A couple of dozen servers running about 50 services each

Agenda:

- Historical introduction
- Technical stuff

Background and Agenda

Big service-oriented backend infrastructure.

- Infrastructure which provides API for our gaming portals
- A couple of dozen servers running about 50 services each

Agenda:

- Historical introduction
- Technical stuff

Don't hesitate to interrupt!

My point today

My point today

Management of big battles is very similar to running distributed systems.

My point today

Management of big battles is very similar to running distributed systems.

I have a good example for you.

Battle of Stalingrad

1942.08.23 – 1943.02.02

Image source: Deutsches Bundesarchiv, RIA Novosti Archive

Rubble battle

Image source: Deutsches Bundesarchiv, RIA Novosti Archive

Peer-to-peer communication

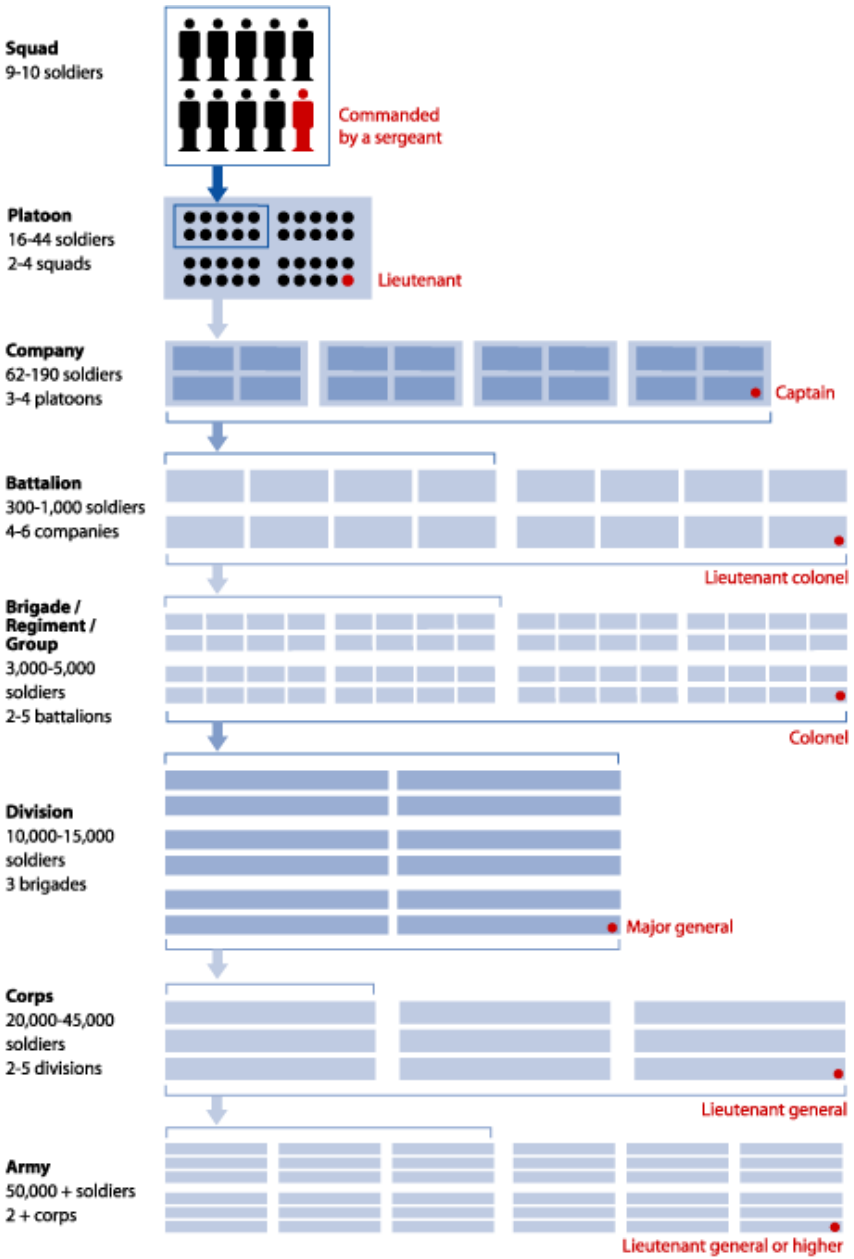
- Impossible to determine location.
- Radio was unreliable and useless.
- Reinforcement/supply requests were just voice.

Heterogeneous

- Infantry
- Tank fleet
- Air Force
- Medical staff
- Commanders
- ...

Image source: `http://navy.mil`

Communication channels



It is clear who gets orders from who. Very clear.

Image source:
`http://www.vetfriends.com/military_structure/`

Dynamic environment

Nature of the battle is dynamic:

Dynamic environment

Nature of the battle is dynamic:
Losses and reinforcements change the dynamics of the battlefield.

Big Plan

It is not enough to only take care of your business.
All units must work to achieve a common goal.

Situation at 1942.11.15

Dire Soviet situation:

- Huge causalities

Situation at 1942.11.15

Dire Soviet situation:

- Huge causalities
- Red Army life expectancy:
  - Soldier: < 1 day
  - Officer: < 3 days

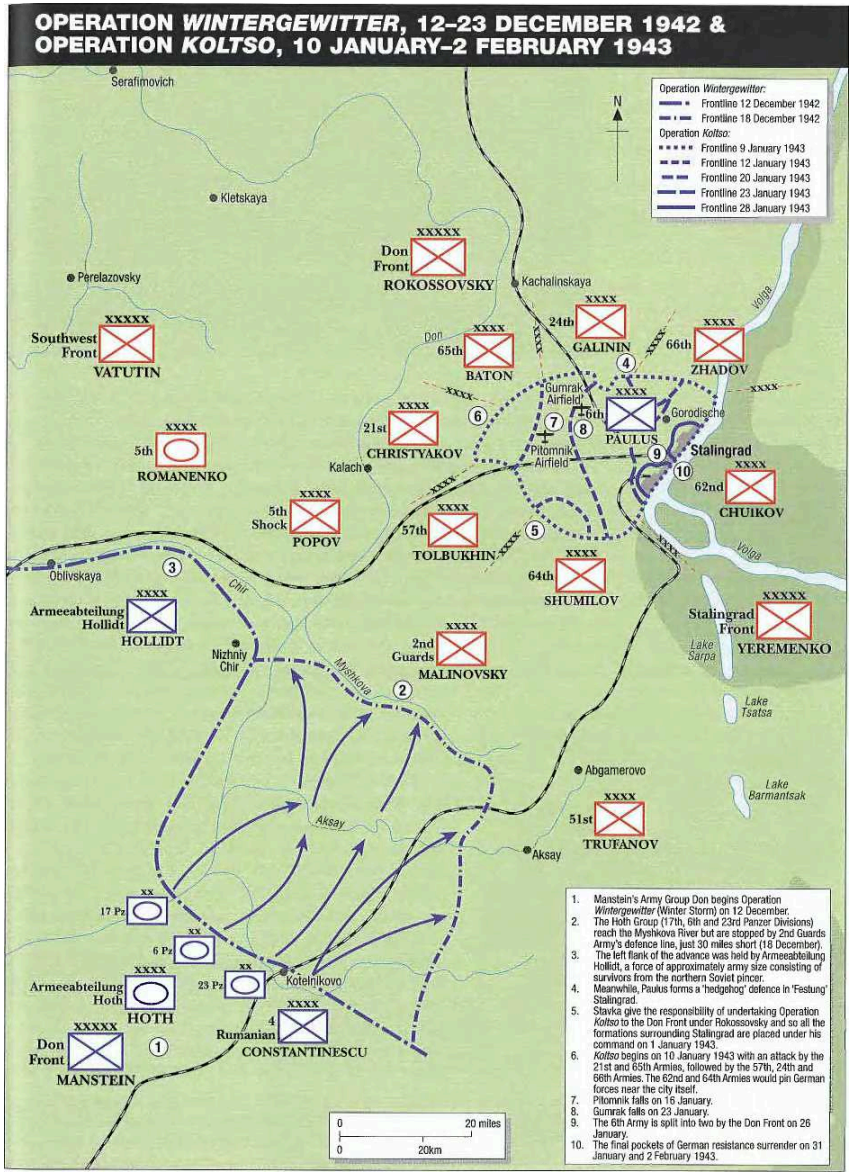
Situation at 1942.11.15

Dire Soviet situation:

- Huge causalities
- Red Army life expectancy:
  - Soldier: < 1 day
  - Officer: < 3 days
- Germans have 90% of the city

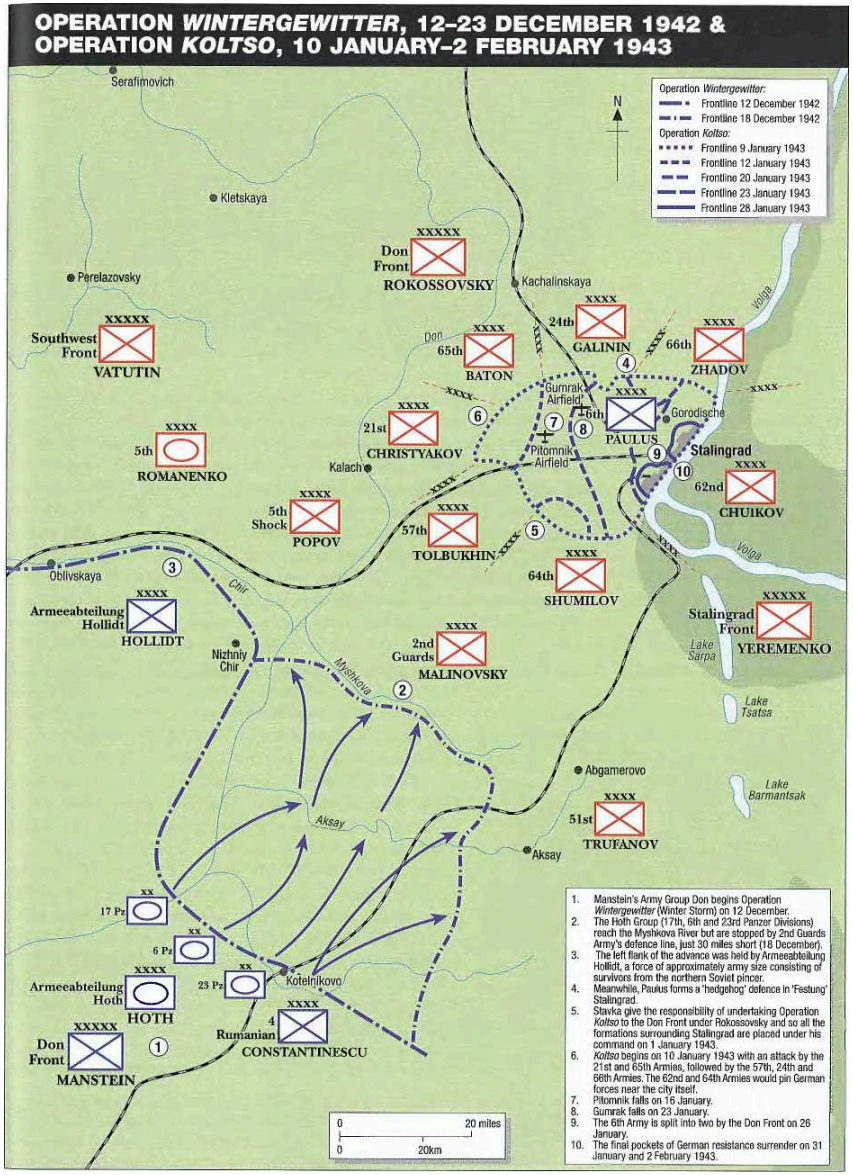
Image source: Antill, P., Dennis, P. *Stalingrad 1942 (Campaign)*. Osprey Publishing (June 19, 2007).

Situation after 1942.11.23



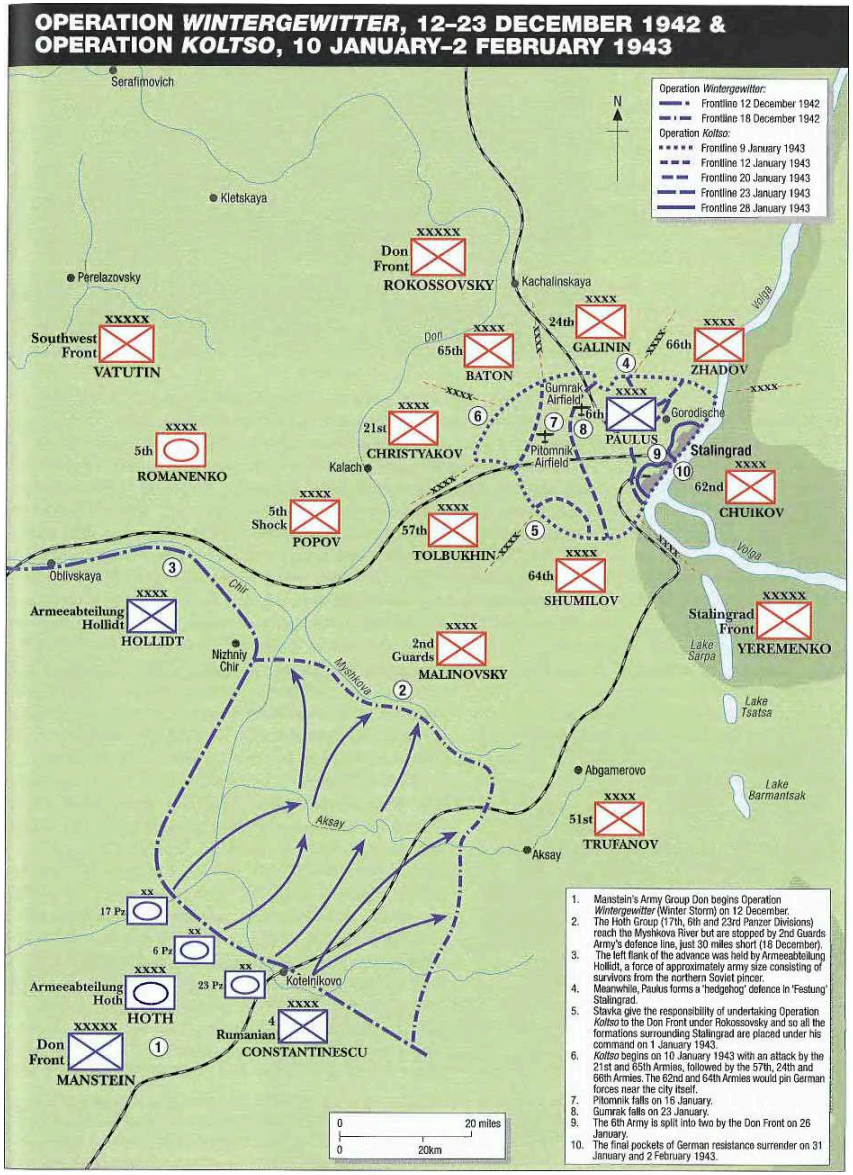
- Germans surrounded by soviets

Situation after 1942.11.23



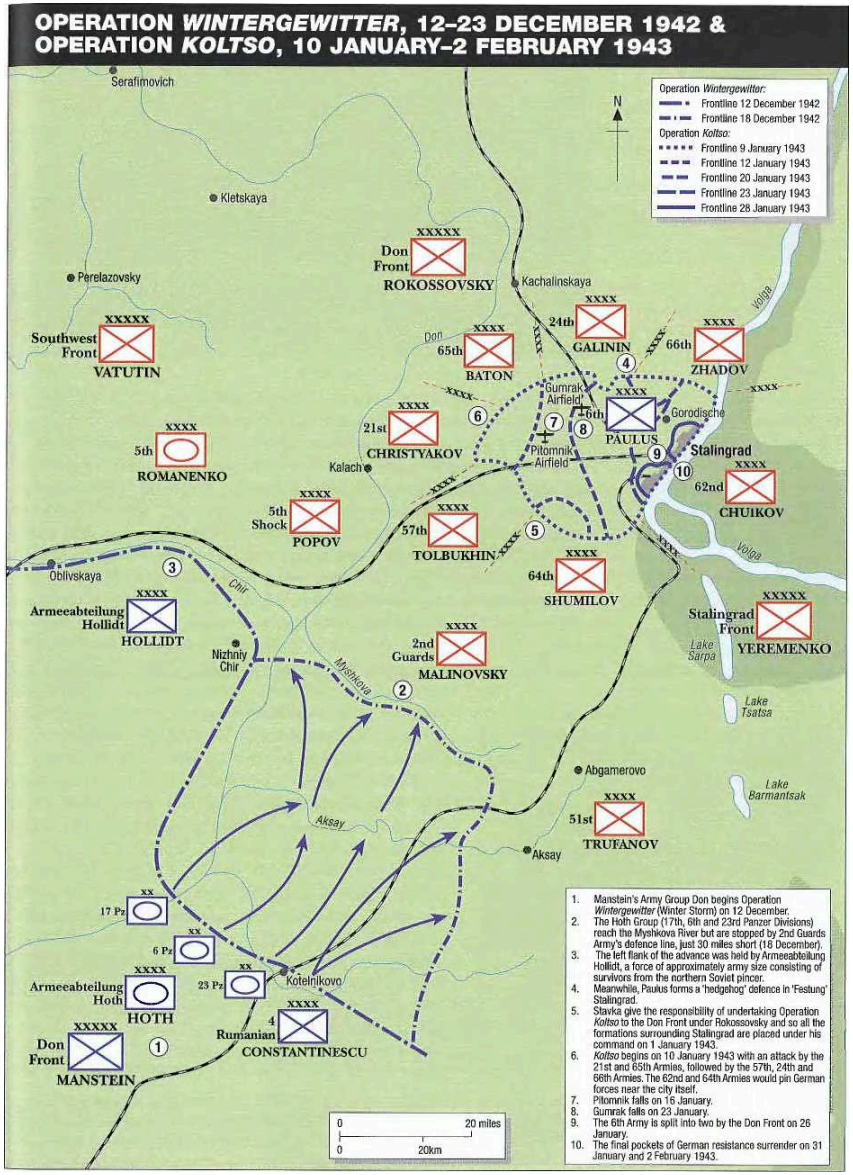
- Germans surrounded by soviets
- City is Germans', Germans' are Soviets'

Situation after 1942.11.23



- Germans surrounded by soviets
- City is Germans', Germans' are Soviets'
- Turning point of the battle

Situation after 1942.11.23



- Germans surrounded by soviets
- City is Germans', Germans' are Soviets'
- Turning point of the battle
- 6'th Army (the surrounded one) was destroyed.

Image source: Antill, P., Dennis, P. *Stalingrad 1942 (Campaign)*. Osprey Publishing (June 19, 2007).

Outline

You sure you have no questions?

Set the grounds

We have a lot of Erlang nodes trying to achieve a common goal.

Set the grounds

We have a lot of Erlang nodes trying to achieve a common goal.

- $\approx$ 50 nodes per server

Set the grounds

We have a lot of Erlang nodes trying to achieve a common goal.

- $\approx$ 50 nodes per server
- a couple of dozens of servers

Set the grounds

We have a lot of Erlang nodes trying to achieve a common goal.

- $\approx$ 50 nodes per server
- a couple of dozens of servers

Example services:

- Higscores

Set the grounds

We have a lot of Erlang nodes trying to achieve a common goal.

- $\approx$ 50 nodes per server
- a couple of dozens of servers

Example services:

- Higscores
- Authentication

Set the grounds

We have a lot of Erlang nodes trying to achieve a common goal.

- $\approx$ 50 nodes per server
- a couple of dozens of servers

Example services:

- Higscores
- Authentication
- Chat

Set the grounds

We have a lot of Erlang nodes trying to achieve a common goal.

- $\approx$ 50 nodes per server
- a couple of dozens of servers

Example services:

- Higscores
- Authentication
- Chat
- User profiles

Set the grounds

We have a lot of Erlang nodes trying to achieve a common goal.

- $\approx$ 50 nodes per server
- a couple of dozens of servers

Example services:

- Higscores
- Authentication
- Chat
- User profiles
- ...

Set the grounds

We have a lot of Erlang nodes trying to achieve a common goal.

- $\approx$ 50 nodes per server
- a couple of dozens of servers

Example services:

- Higscores
- Authentication
- Chat
- User profiles
- ...

How to connect them?

Peer-to-peer communication

You don't want bottlenecks.
You don't want single points of failure.

Dynamic nodes

Nodes and services start and stop all the time.

Dynamic nodes

Nodes and services start and stop all the time.
The system must continue to function and self-heal.

Partially connected network

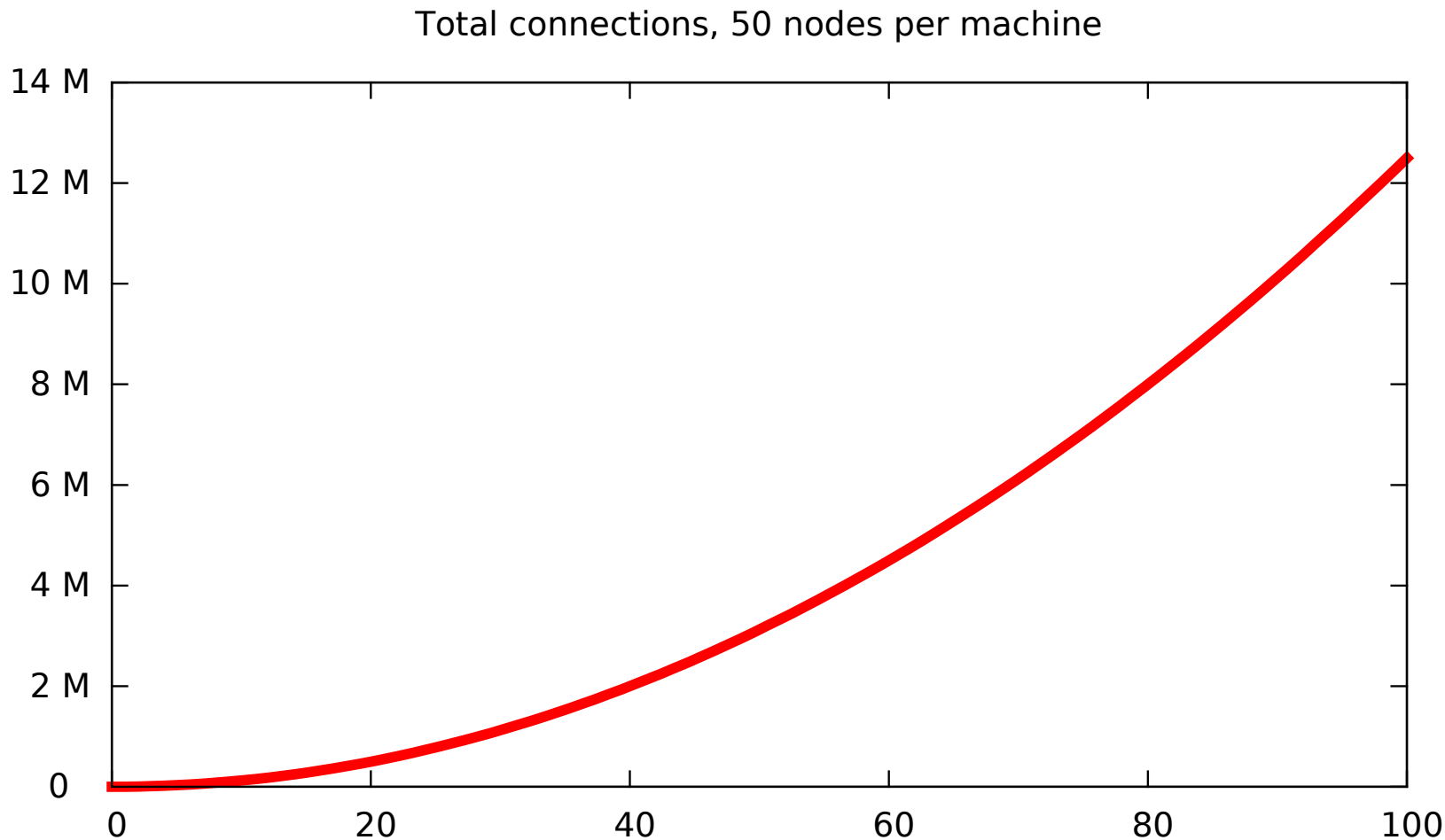
n : number of nodes.

Total connections $= \frac{n(n-1)}{2}$

Partially connected network

n : number of nodes.

Total connections $= \frac{n(n-1)}{2}$



Conclusions (for now)

It pays off to optimize the topology so communication is more effective, e.g.:

- Army
- Software Defined Networking
- Management

Remember?

Erlang	**Battle**
P2P communication	P2P communication
Multi app groups	Heterogeneous
Dynamic nodes	Dynamic environment
Partially connected network	Communication channels

Remember?

Erlang	**Battle**
P2P communication	P2P communication
Multi app groups	Heterogeneous
Dynamic nodes	Dynamic environment
Partially connected network	Communication channels

Maintaining a distributed system is like managing a battle.

Remember?

Erlang	**Battle**
P2P communication	P2P communication
Multi app groups	Heterogeneous
Dynamic nodes	Dynamic environment
Partially connected network	Communication channels

Maintaining a distributed system is like managing a battle.

We can be generals.

We need to solve this

What	**spapi-router**	**pg2**	**gproc**
P2P communication	✓	✓	✓
Multi app groups	✓	✓	✓
Dynamic nodes	✓	✓	✓
Partially connected network by limiting connections	✓	x	x

spapi-router features in a nutshell

A library.

- Creates a mesh network

  Connects (hidden) nodes like configured

spapi-router features in a nutshell

A library.

- Creates a mesh network

  Connects (hidden) nodes like configured
- Abstracts destination

  RPC-call a service, not a node

spapi-router features in a nutshell

A library.

- Creates a mesh network

  Connects (hidden) nodes like configured
- Abstracts destination

  RPC-call a service, not a node
- Mature and optimized

  Used for 2,5 years in a sufficiently large SOA

spapi-router features in a nutshell

A library.

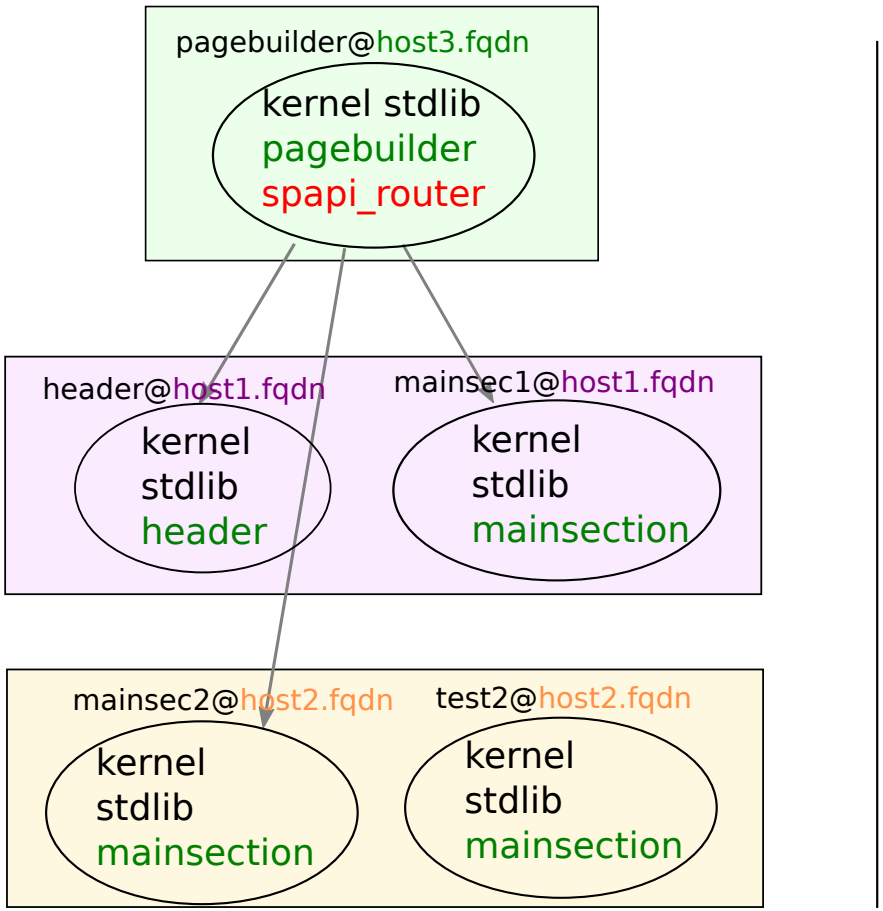
- Creates a mesh network

  Connects (hidden) nodes like configured
- Abstracts destination

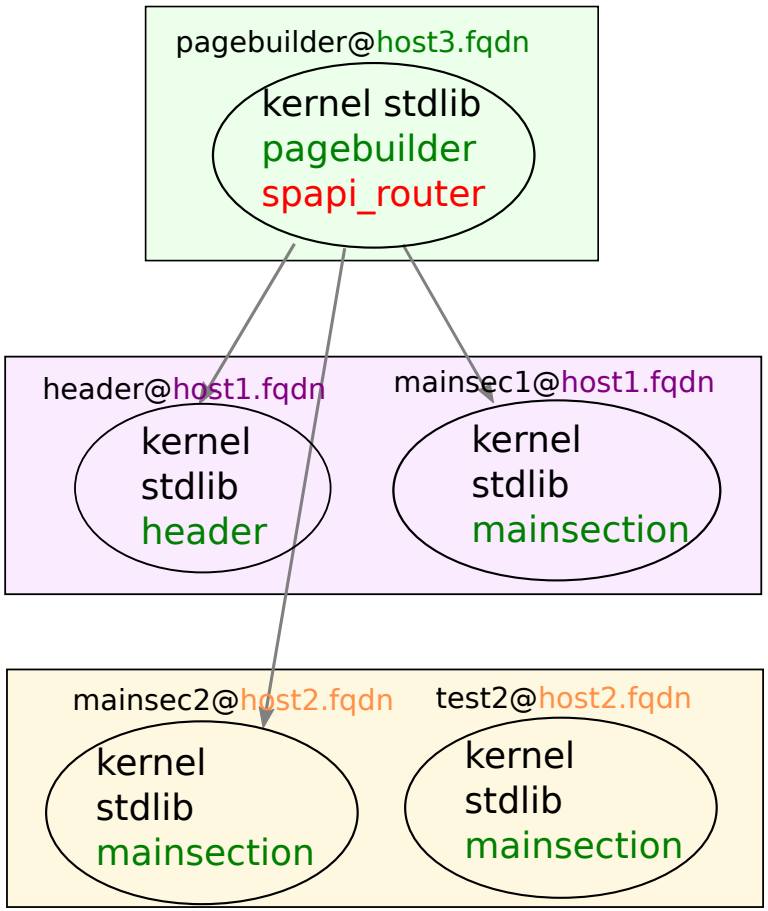
  RPC-call a service, not a node
- Mature and optimized

  Used for 2,5 years in a sufficiently large SOA
- Instrumented

For example


pagebuilder@host3.fqdn
kernel stdlib
pagebuilder
spapi_router
header@host1.fqdn
kernel
stdlib
header
mainsec1@host1.fqdn
kernel
stdlib
mainsection
mainsec2@host2.fqdn
kernel
stdlib
mainsection
test2@host2.fqdn
kernel
stdlib
mainsection

For example



Configuration of pagebuilder@host3.fqdn:

```
{spapi_router, [
  {host_names, [
    "host1.fqdn",
    "host2.fqdn",
    ]},
  {workers, [
      {"^header[0-9]*", [header]},
      {"^mainsec[0-9]*", [mainsection]},
    ]}
  ]}
```

More configuration

- `hosts_monitor_interval_ms`
- `world_monitor_interval_ms`
- `worker_monitor_interval_ms`
- `callback_module`

How it works

```
{spapi_router, [
  {host_names, [
    "host1.fqdn",
    "host2.fqdn",
    ]},
  {workers, [
      {"^header[0-9]*", [header]},
      {"^mainsec[0-9]*", [mainsection]},
    ]}
  ]}
```

How it works

```
{spapi_router, [
  {host_names, [
    "host1.fqdn",
    "host2.fqdn",
    ]},
  {workers, [
      {"^header[0-9]*", [header]},
      {"^mainsec[0-9]*", [mainsection]},
    ]}
  ]}
```

1. Connects to `host_names`

How it works

```
{spapi_router, [
  {host_names, [
    "host1.fqdn",
    "host2.fqdn",
    ]},
  {workers, [
      {"^header[0-9]*", [header]},
      {"^mainsec[0-9]*", [mainsection]},
    ]}
  ]}
```

1. Connects to `host_names`
2. Asks EPMD for running nodes

How it works

```
{spapi_router, [
  {host_names, [
    "host1.fqdn",
    "host2.fqdn",
    ]},
  {workers, [
      {"^header[0-9]*", [header]},
      {"^mainsec[0-9]*", [mainsection]},
    ]}
  ]}
```

1. Connects to `host_names`
2. Asks EPMD for running nodes
3. Connects to nodes matching regexp

How it works

```
{spapi_router, [
  {host_names, [
    "host1.fqdn",
    "host2.fqdn",
    ]},
  {workers, [
      {"^header[0-9]*", [header]},
      {"^mainsec[0-9]*", [mainsection]},
    ]}
  ]}
```

1. Connects to `host_names`
2. Asks EPMD for running nodes
3. Connects to nodes matching regexp
4. Checks for applications in nodes

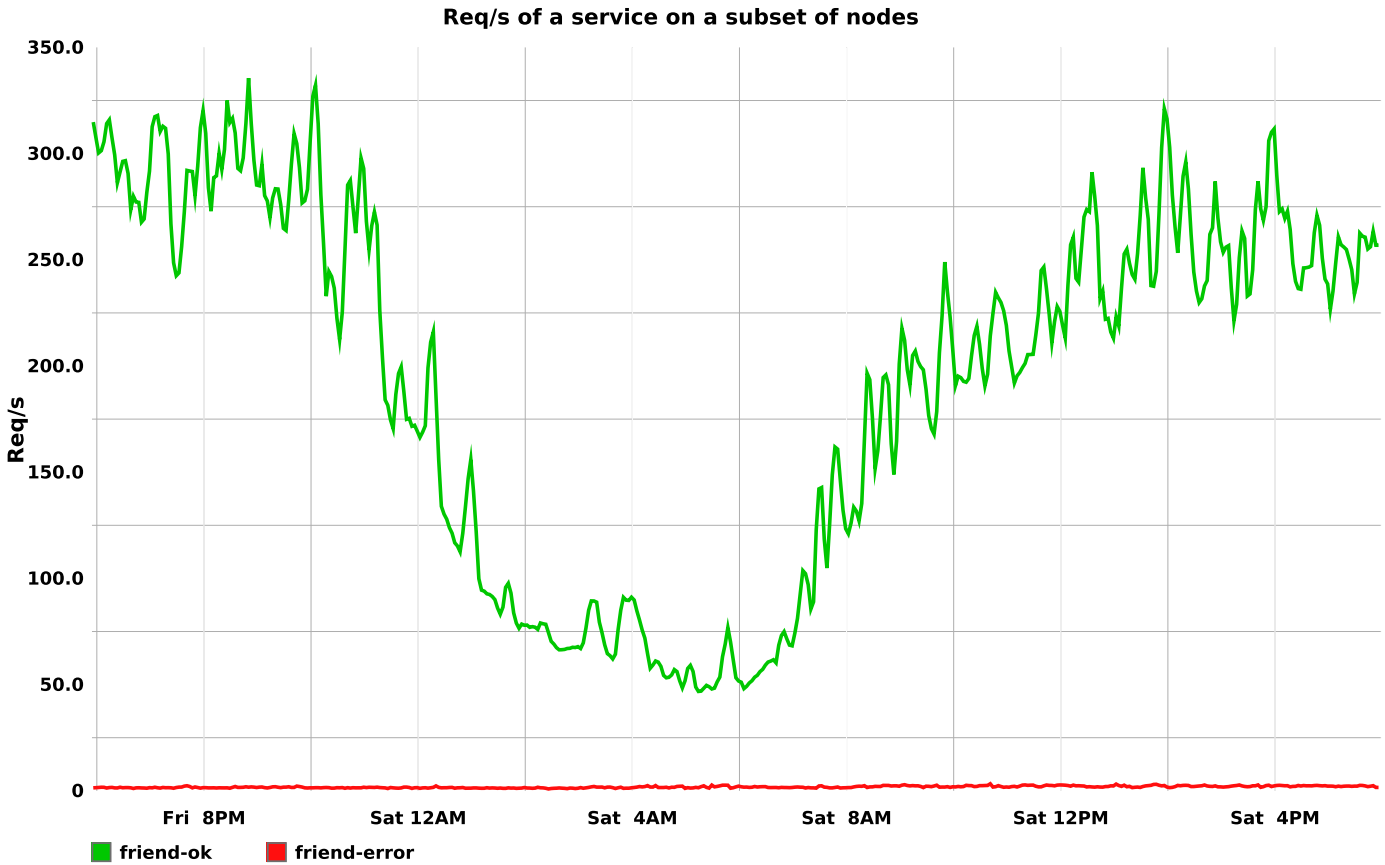
How it works

```
{spapi_router, [
  {host_names, [
    "host1.fqdn",
    "host2.fqdn",
    ]},
  {workers, [
      {"^header[0-9]*", [header]},
      {"^mainsec[0-9]*", [mainsection]},
    ]}
  ]}
```

1. Connects to `host_names`
2. Asks EPMD for running nodes
3. Connects to nodes matching regexp
4. Checks for applications in nodes
5. Connects to relevant nodes

Why instrument?

Helps understand the system is sound.



The first thing you want to instrument is the border of your service.

Effortless instrumentation for all calls via spapi-router.

callback_module #1

```
%% New resource is detected
-callback new_resource({Service :: atom(), node()},
                       opts()) -> any().

%% Existing resource is lost
%% (node disconnect, shutdown, etc).
-callback lost_resource({Service :: atom(), node()},
                        opts()) -> any().
```

callback_module #2

```
-type log_spec() :: {
    Service :: atom(),
    Module :: atom(),
    Function :: atom()
}.

%% Time instrumentation
-callback measure(log_spec(),
                  fun(() -> A), opts()) -> A.

%% Called on success/failure of a function call.
-callback success(log_spec(), opts()) -> term().
-callback failure(log_spec(), opts()) -> term().
```

Calling others

Calling others

- `spr_router:call(piqi_rpc, erlang, node, []).`

Calling others

- `spr_router:call(piqi_rpc, erlang, node, []).`
- `spr_router:call_all(piqi_rpc, erlang, node, []).`

Calling others

- `spr_router:call(piqi_rpc, erlang, node, []).`
- `spr_router:call_all(piqi_rpc, erlang, node, []).`

Extras:

- `call/5`
- `call_all/5`
- `list_workers/0`
- `list_workers/1`
- `list_hosts/0`

Future optimizations

- Takes time to figure out a 'stop'.
- Monitor `application_controller` instead of node.
- One node == one service.

How to change nodes?

Puppet plus

- RelUp
- ... or anything really:

```
spr_app:config_change([], [], []).
```

Battle stories

Battle stories

- Tried to disconnect from irrelevant nodes first

Battle stories

- Tried to disconnect from irrelevant nodes first
  - what if relationship is one-way?

Battle stories

- Tried to disconnect from irrelevant nodes first
  - what if relationship is one-way?
  - one misbehaving component can bring the system down

Battle stories

- Tried to disconnect from irrelevant nodes first
  - what if relationship is one-way?
  - one misbehaving component can bring the system down
- Fully connected network experience ($>$ 1K nodes)

Battle stories

- Tried to disconnect from irrelevant nodes first
  - what if relationship is one-way?
  - one misbehaving component can bring the system down
- Fully connected network experience ($>$ 1K nodes)
  - everyone should try that

Battle stories

- Tried to disconnect from irrelevant nodes first
  - what if relationship is one-way?
  - one misbehaving component can bring the system down
- Fully connected network experience ($>$ 1K nodes)
  - everyone should try that
  - thanks to off-peak and 10G NIC

Stats

2012-01-12 Initial commit (Thijs Terlouw)

Stats

2012-01-12 Initial commit (Thijs Terlouw)

2012-02-16 `1.0.0` – first prod (Thijs Terlouw)

Stats

2012-01-12 Initial commit (Thijs Terlouw)

2012-02-16 `1.0.0` – first prod (Thijs Terlouw)

2012-08-24 `1.2.0` – broadcast (Enrique Paz)

Stats

2012-01-12 Initial commit (Thijs Terlouw)

2012-02-16 `1.0.0` – first prod (Thijs Terlouw)

2012-08-24 `1.2.0` – broadcast (Enrique Paz)

2014-06-20 `2.1.0` – many contributions by Spil Games

Stats

2012-01-12 Initial commit (Thijs Terlouw)

2012-02-16 `1.0.0` – first prod (Thijs Terlouw)

2012-08-24 `1.2.0` – broadcast (Enrique Paz)

2014-06-20 `2.1.0` – many contributions by Spil Games

2014-07-04 Public pre-release

```
commit b3b7aad9ca14ed230f28635826b371b6bbea3840
Author: Motiejus Jakštys <motiejus.jakstys@spilgames.com>
Date:   Wed Jun 4 14:39:40 2014 +0200

    Initial commit

    21 files changed, 2984 insertions(+)
```

Stats

2012-01-12 Initial commit (Thijs Terlouw)

2012-02-16 `1.0.0` – first prod (Thijs Terlouw)

2012-08-24 `1.2.0` – broadcast (Enrique Paz)

2014-06-20 `2.1.0` – many contributions by Spil Games

2014-07-04 Public pre-release

```
commit b3b7aad9ca14ed230f28635826b371b6bbea3840
Author: Motiejus Jakštys <motiejus.jakstys@spilgames.com>
Date:   Wed Jun 4 14:39:40 2014 +0200

    Initial commit

    21 files changed, 2984 insertions(+)
```

2014-07-10 `http://github.com/spilgames/spapi-router`

Outline

You sure you have no questions?

QA