The C1K Problem -
Connecting Thousands of Erlang Devices

Peer Stritzinger, Erlang User Conference
Stockholm, June 2015

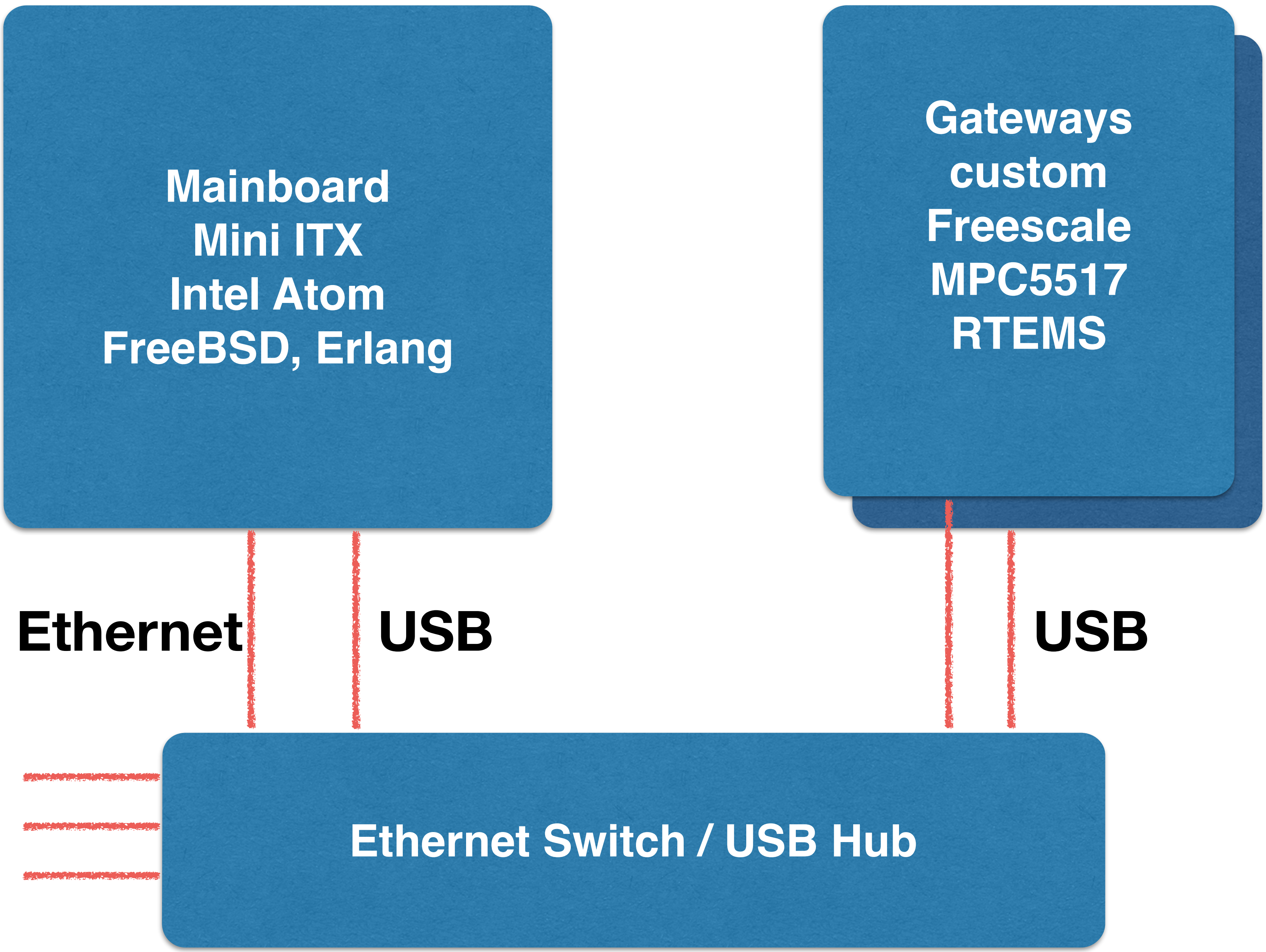
DIPL. PHYS. PEER STRITZINGER GMBH


hydraprog3
hydraprog3


hydraprog3


hydraprog3


X1
X2
HP3 GW-LCFM
X3

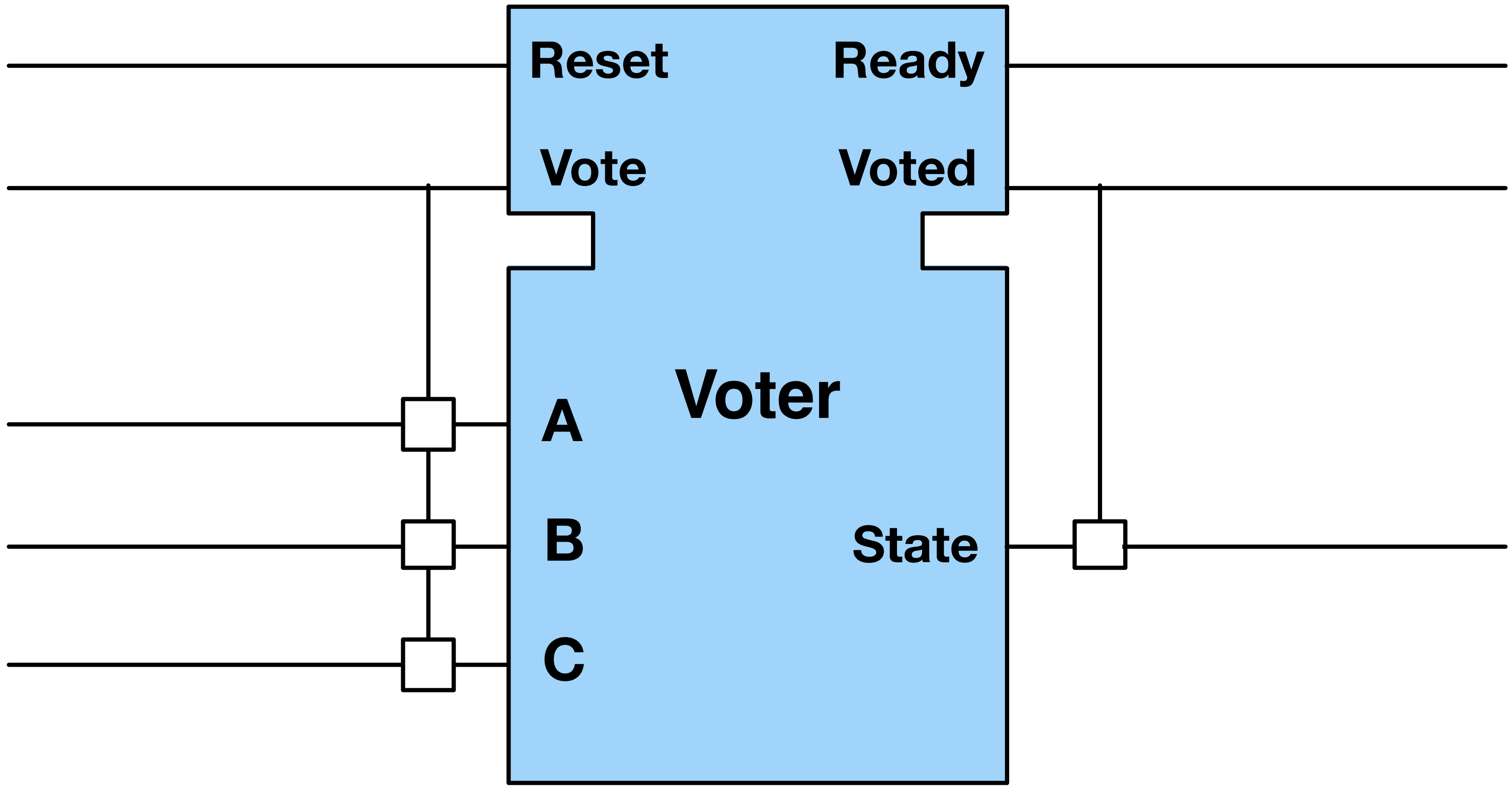


DIPL. PHYS. PEER STRITZINGER GMBH

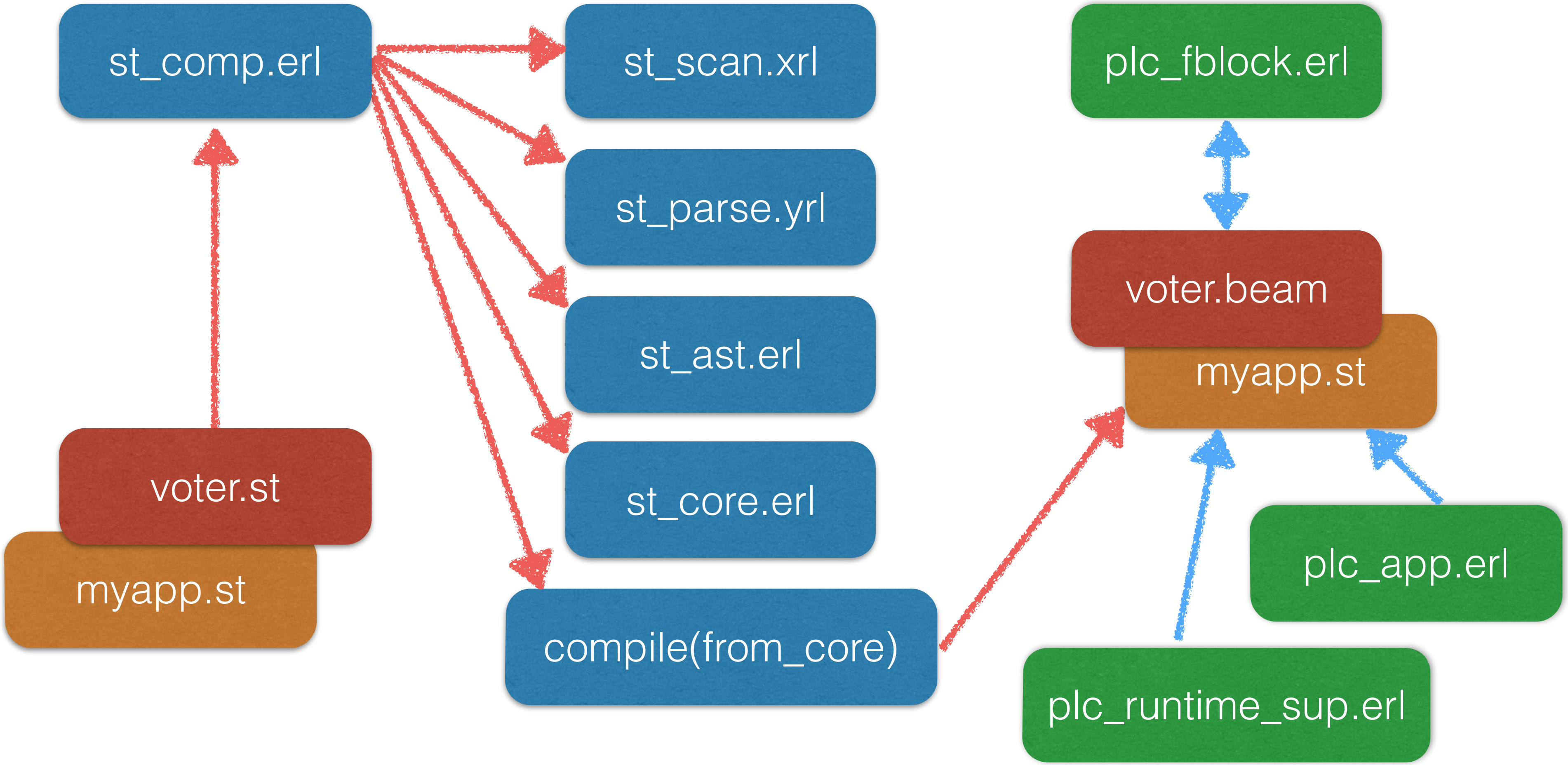
PLC

DIPL. PHYS. PEER STRITZINGER GMBH

Distributed PLC with IEC61499

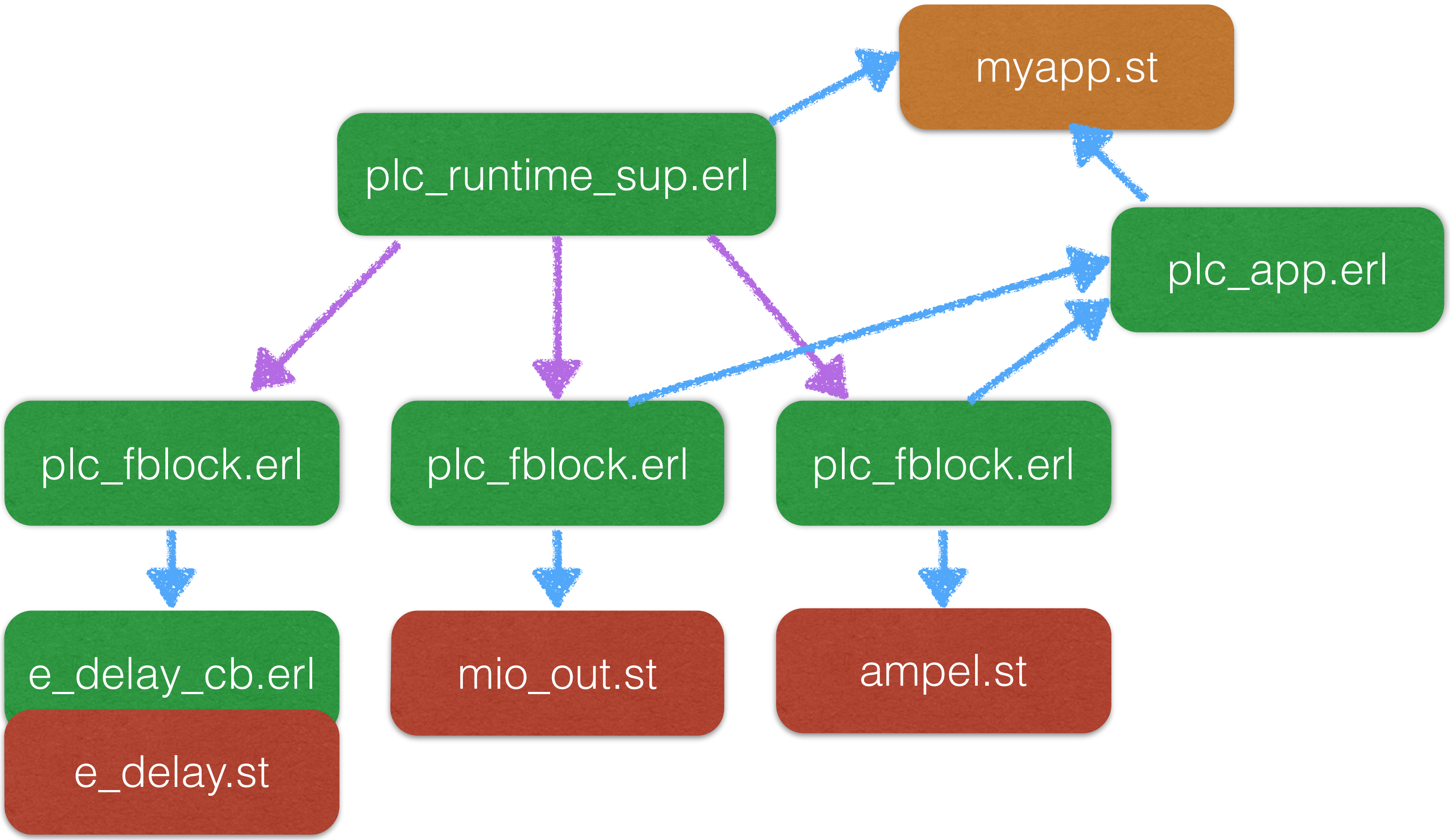


IEC61499 Compiler



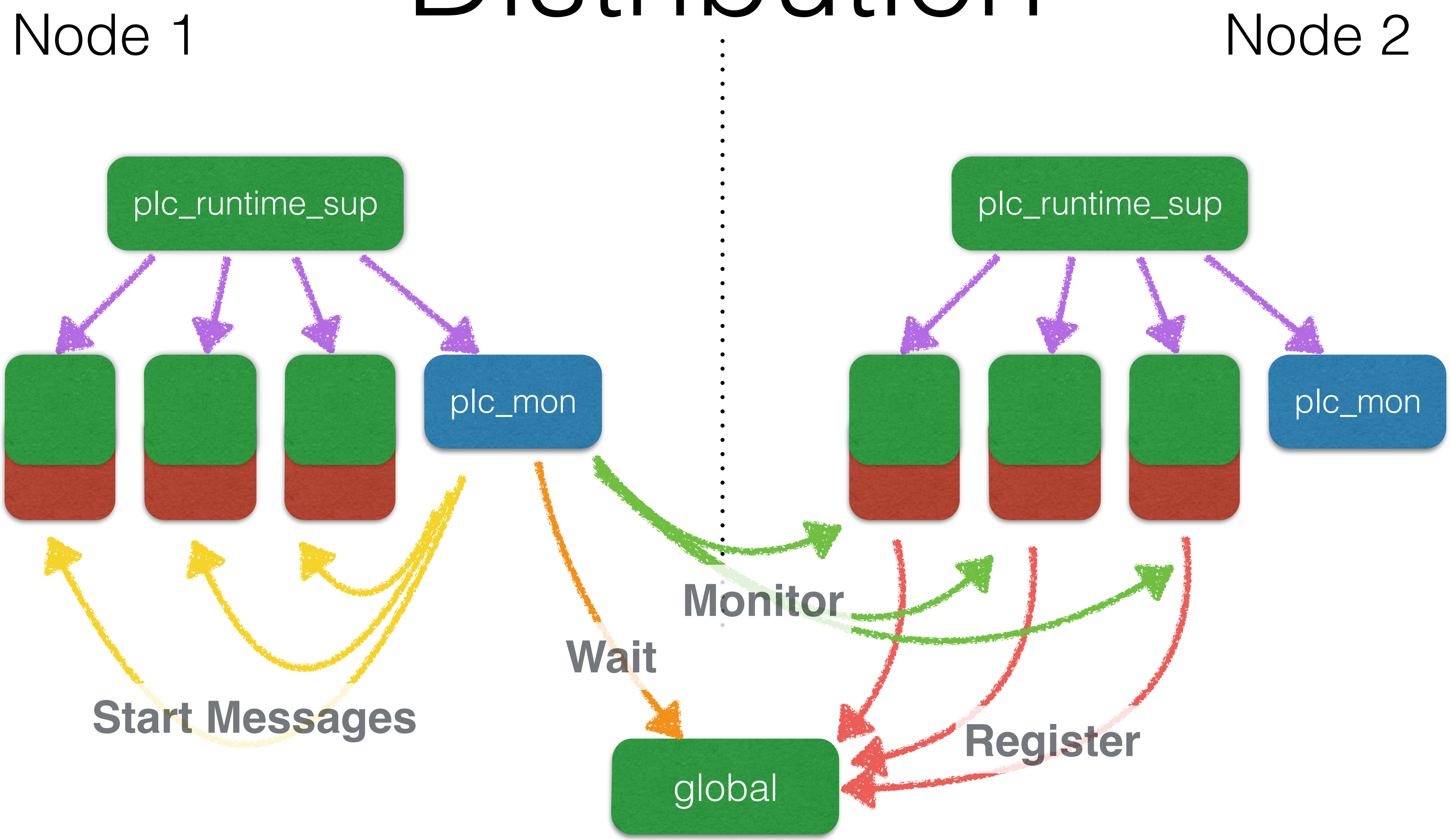
DIPL. PHYS. PEER STRITZINGER GMBH

IEC61499 Runtime



DIPL. PHYS. PEER STRITZINGER GMBH

Distribution



DIPL. PHYS. PEER STRITZINGER GMBH

ERLANG

\+ Embedded With RTEMS www.rtems.com

= GRISP

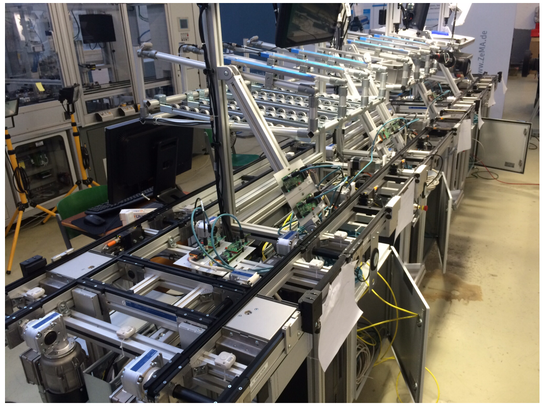
www.grisp.org

Application

- Next Generation RFID System
- Industry 4.0, Smart Factory System
- Distributed data sharing and material routing in Erlang
- Programmability by PLC Programmers
- Research Project: Cyber-physical IT-Systems to handle the complexity of a new Generation of multi adaptive Factories

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung
DLR
PT-SW des BMBF


24 Vout
24 Vin
ETH 3
ETH 2
ETH 1
PWR/SYS
ETH 3
ETH 2
ETH 1
Rexroth
Bosch Group
ATC/8I0-1HF
3 842 410 506
RFID
137
110




CeBIT
future talk
CeBIT
Intelligente Cyber-physische IT-Systeme


Rexroth

DIPL. PHYS. PEER STRITZINGER GMBH

DIPL. PHYS. PEER STRITZINGER GMBH

Scaling

- Registering everything in global registry is slow
  - at least if there is latency like in a real network
  - Startup of 15 nodes takes 10-15 minutes
- Fully connected Erlang distribution will break at some point between 15 and 1000 nodes

Benchmarking lesson

- Many distributed algorithm performance are very sensitive to bandwidth and latency
- Application with 13 nodes on 13 embedded systems with 100Mbit network startup > 10 **minutes**
- Same application with13 Erlang nodes on laptop via localhost startup < 10 **seconds**
- Very important to closely control latency/bandwidth for benchmarking!

Possible solutions

- SD Erlang
- gproc - locks_leader branch
- Hidden nodes
- Fixing Erlang distribution

SD Erlang

- http://www.dcs.gla.ac.uk/research/sd-erlang/
- Overlapping groups of nodes
- Global module like name registry limited to group

DIPL. PHYS. PEER STRITZINGER GMBH

gproc / locks_leader

- Distributed gproc is known to have issues due to gen_leader
- There is a new branch using locks_leader
- https://github.com/uwiger/gproc/tree/uw-locks_leader
- Supposed to be faster than global
- Still fully connected network

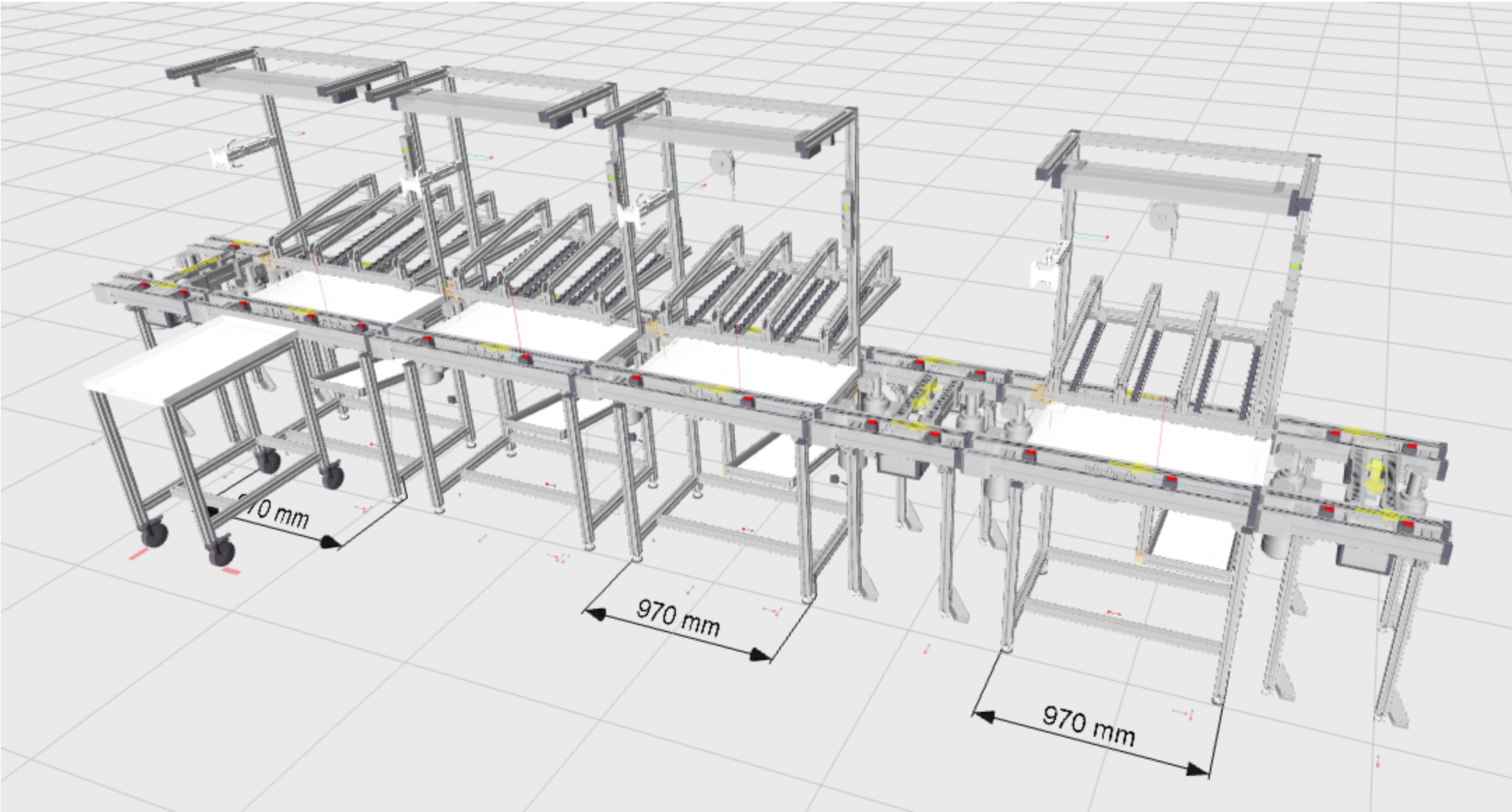
Hidden nodes

- Connect only when communicated to
- „Invisible“ to to normal nodes() so won’t get connected by global e.al.
- Start hidden node: **erl -hidden**
- Use **nodes/1** function to query
- Connected hidden nodes can be monitored

Hidden Nodes (cont.)

- erlang:send/3 option noconnect
- Sending to {*name*, *node*}
- All of net_kernel

Closer look at the application



DIPL. PHYS. PEER STRITZINGER GMBH

Closer look at the application (cont)

atc0124

o3: us1_hqe2_down
o2: us1_hqe2_up
o1: us1_ve2
i5: us1_wip2
i6: us1_qst2
i3: us1_b1_ve2
i4: us1_b2_ve2

atc0123

i1: st3_fu_km2_ok
i6: st1_bst
o3: st1_km2_vor

atc0126

i1: st2_fu_km2_ok
i6: st2_bst
o3: st2_km2_vor

atc0132

i1: st3_fu_km2_ok
i6: st3_bst
o3: st3_km2_vor

atc0101

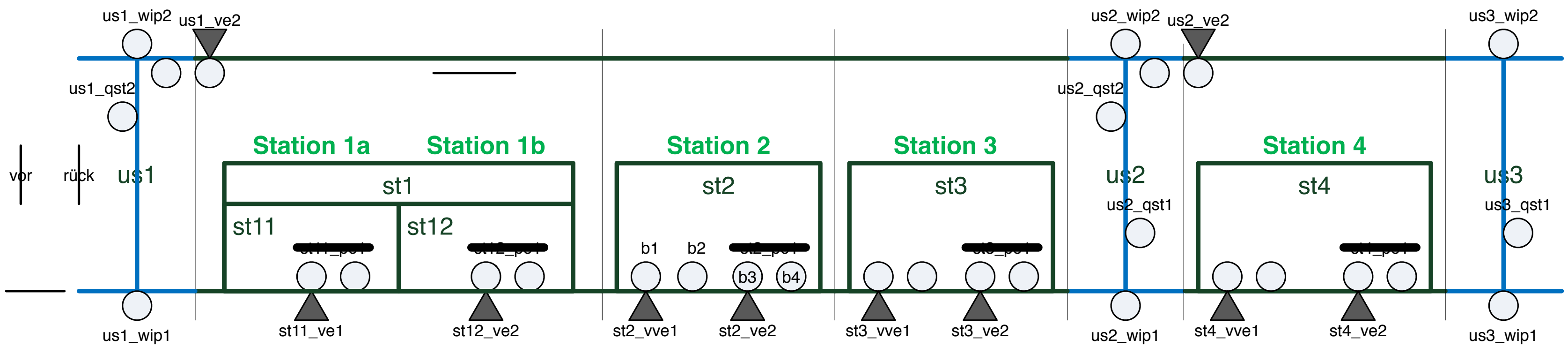
o3: us2_hqe2_down
o2: us2_hqe2_up
o1: us2_ve2
i5: us2_wip2
i6: us2_qst2
i3: us2_b1_ve2
i4: us2_b2_ve2

atc0119

i1: st4_fu_km2_ok
i6: st4_bst
o3: st4_km2_vor

atc0128

o3: us3_hqe2_dov
o2: us3_hqe2_up
i5: us3_wip2



atc0115

o7: us1_km2_vor
o5: us1_km1_vor
o3: us1_hqe1_down
o2: us1_hqe1_up
i5: us1_wip1

i1: us1_fu_km1_ok
i2: us1_fu_km2_ok
i6: air_press_ok

atc0121

o1: st11_ve1
o2: st11_qu1_led
o3: st1_km1_vor
o5: st11_pe1_start
o6: st11_pe1_open
o7: st11_pe1_close

i1: st1_fu_km1_ok
i2: st11_qu1
i3: st11_b3_ve1
i4: st11_b4_ve1
i7: st11_pe1_ready
i8: st11_pe1_most

co111: st11_terminate
di110: st11_wt_present

atc0137

o1: st12_ve2
o2: st12_qu2_led
o5: st12_pe2_start
o6: st12_pe2_open
o7: st12_pe2_close

i2: st12_qu2
i3: st12_b3_ve2
i4: st12_b4_ve2
i7: st12_pe2_ready
i8: st12_pe2_most

co111: st12_terminate
di110: st12_wt_present

atc0134

o1: st2_vve1
o2: st2_s1_qu_led
o3: st2_km1_vor
o5: st2_pe1_start
o6: st2_pe1_open
o7: st2_pe1_close
o8: st2_ve2

i1: st2_fu_km1_ok
i2: st2_qu1
i3: st2_b1_vve1
i4: st2_b2_vve1
i5: st2_b3_ve2
i6: st2_b4_ve2
i7: st2_pe1_ready
i8: st2_pe1_most

co111: st2_terminate
di110: st2_wt_present

atc0110

o1: st3_vve1
o2: st3_s1_qu_led
o3: st3_km1_vor
o5: st3_pe1_start
o6: st3_pe1_open
o7: st3_pe1_close
o8: st3_ve2

i1: st3_fu_km1_ok
i2: st3_qu1
i3: st3_b1_vve1
i4: st3_b2_vve1
i5: st3_b3_ve2
i6: st3_b4_ve2
i7: st3_pe1_ready
i8: st3_pe1_most

co111: st3_terminate
di110: st3_wt_present

atc0139

o8: us2_km2_rck
o7: us2_km2_vor
o5: us2_km1_vor
o3: us2_hqe1_down
o2: us2_hqe1_up

i1: us2_fu_km1_ok
i2: us2_fu_km2_ok
i5: us2_wip1
i6: us2_qst1

atc0117

o1: st4_vve1
o2: st4_s1_qu_led
o3: st4_km1_vor
o5: st4_pe1_start
o6: st4_pe1_open
o7: st4_pe1_close
o8: st4_ve2

i1: st4_fu_km1_ok
i2: st4_qu
i3: st4_b1_vve1
i4: st4_b2_vve1
i5: st4_b3_ve2
i6: st4_b4_ve2
i7: st4_pe1_ready
i8: st4_pe1_most

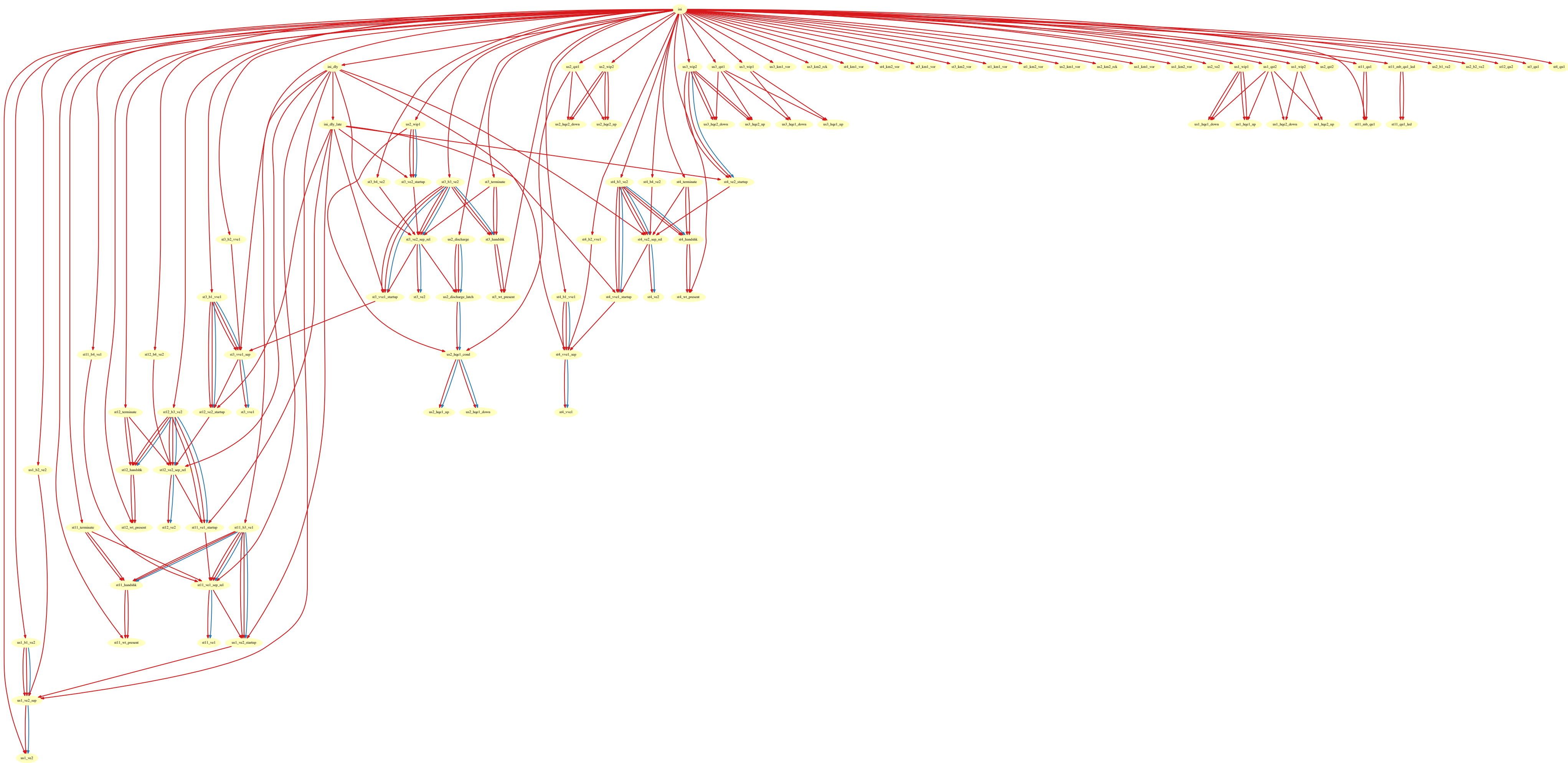
co111: st4_terminate
di110: st4_wt_present

atc0102

o5: us3_km1_vor
o8: us3_km2_rck
o3: us3_hqe1_dov
o2: us3_hqe1_up

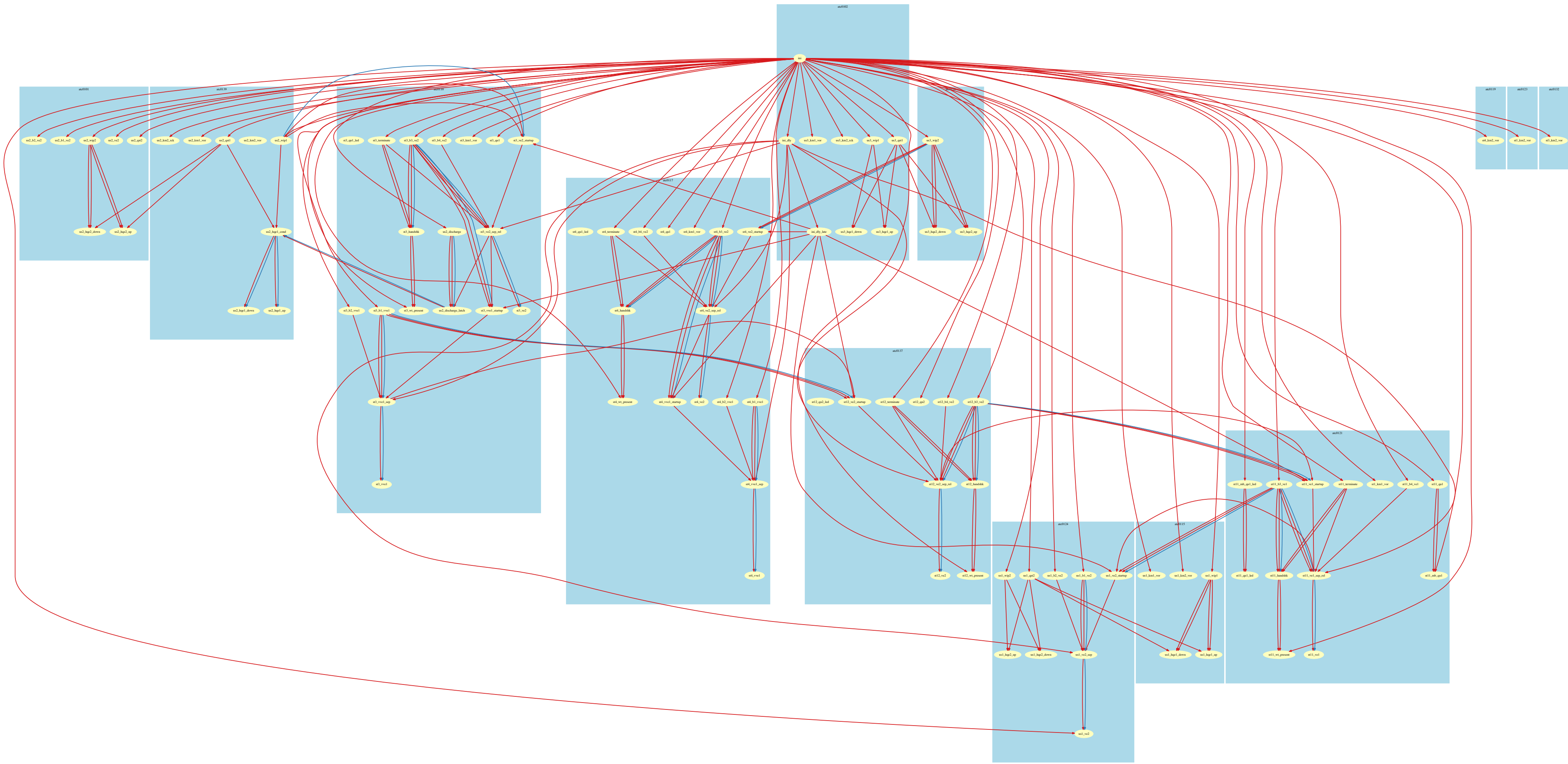
i1: us3_fu_km1_ol
i2: us3_fu_km2_ol
i5: us3_wip1
i6: us3_qst1

Process connections



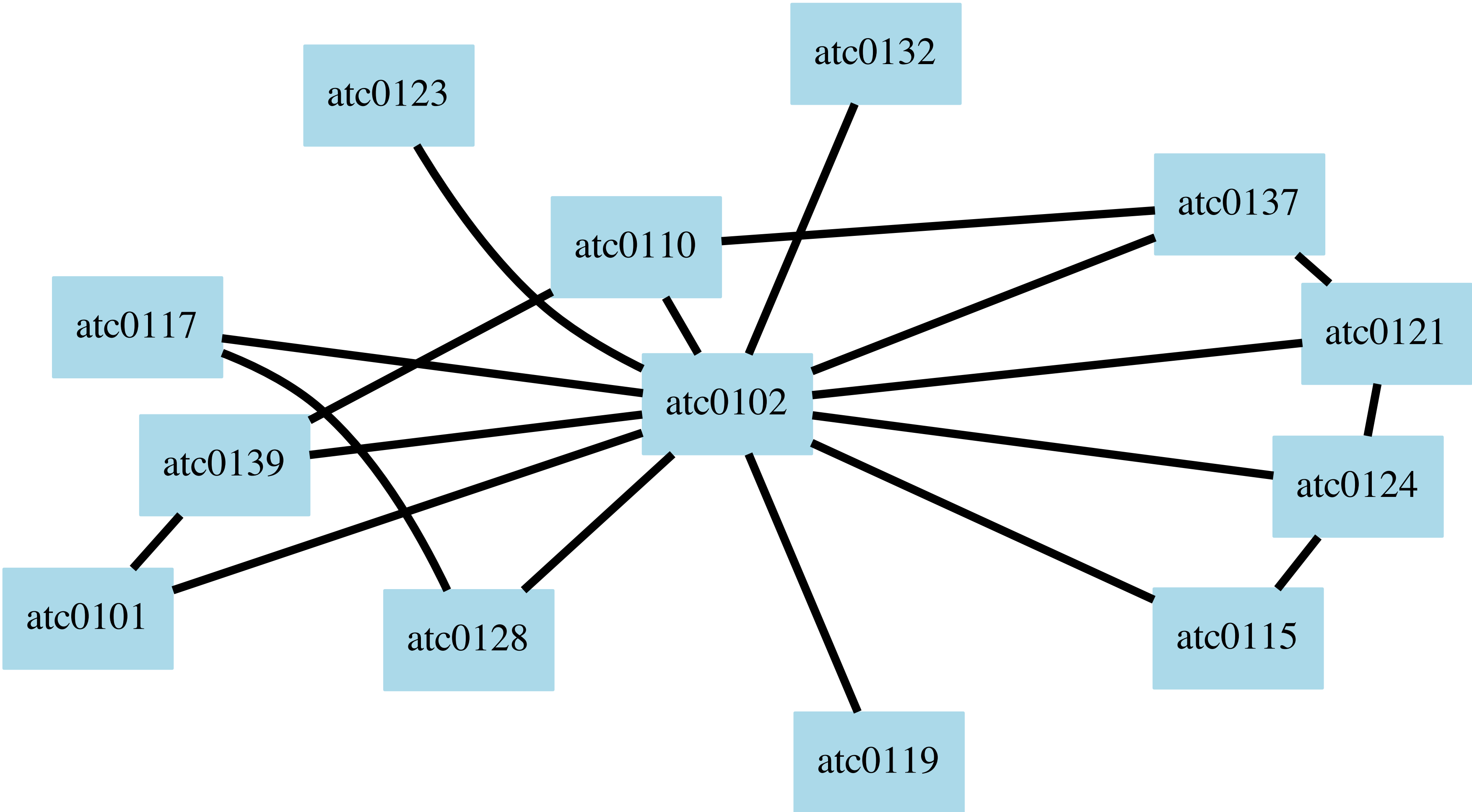
DIPL. PHYS. PEER STRITZINGER GMBH

Processes and Nodes



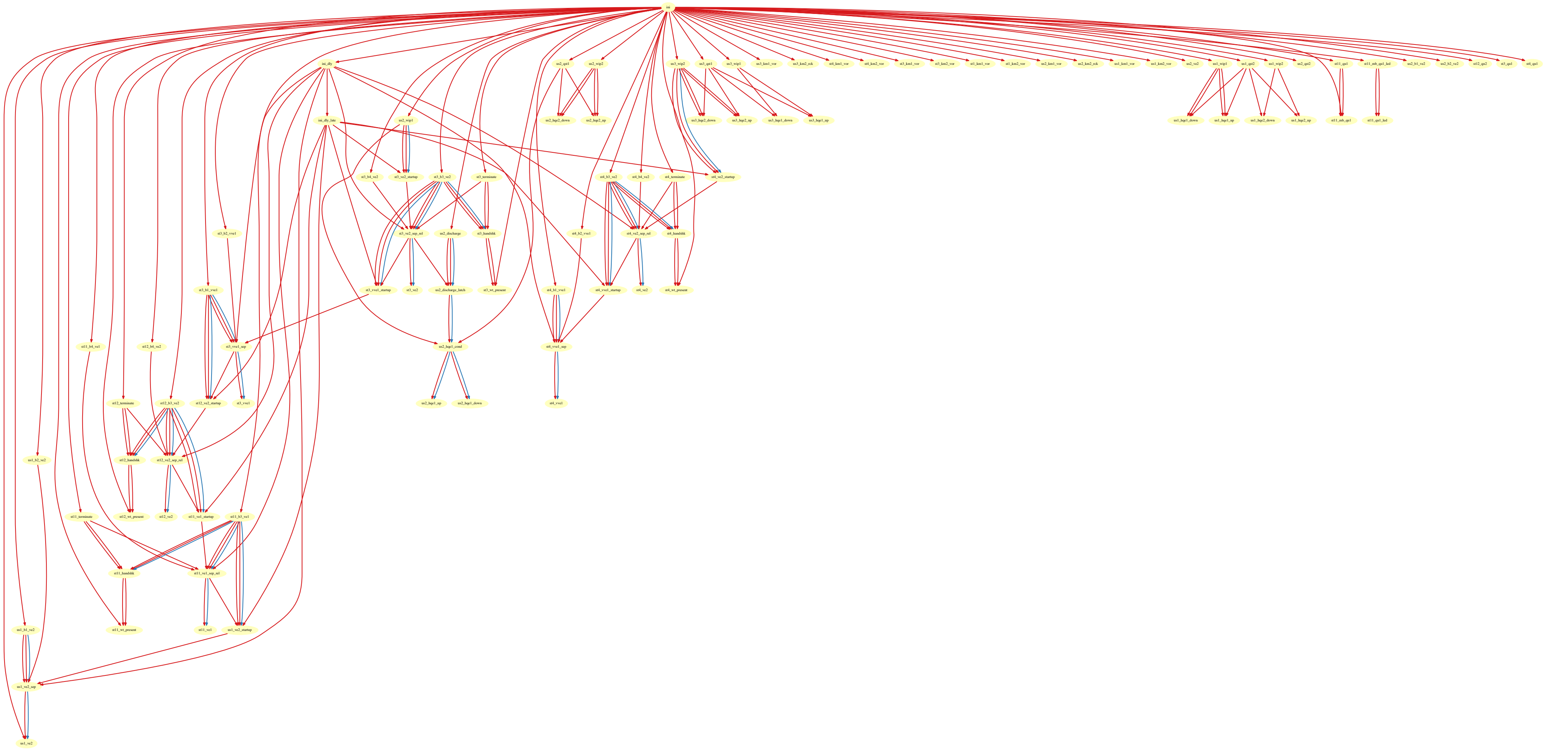
DIPL. PHYS. PEER STRITZINGER GMBH

Node Connections



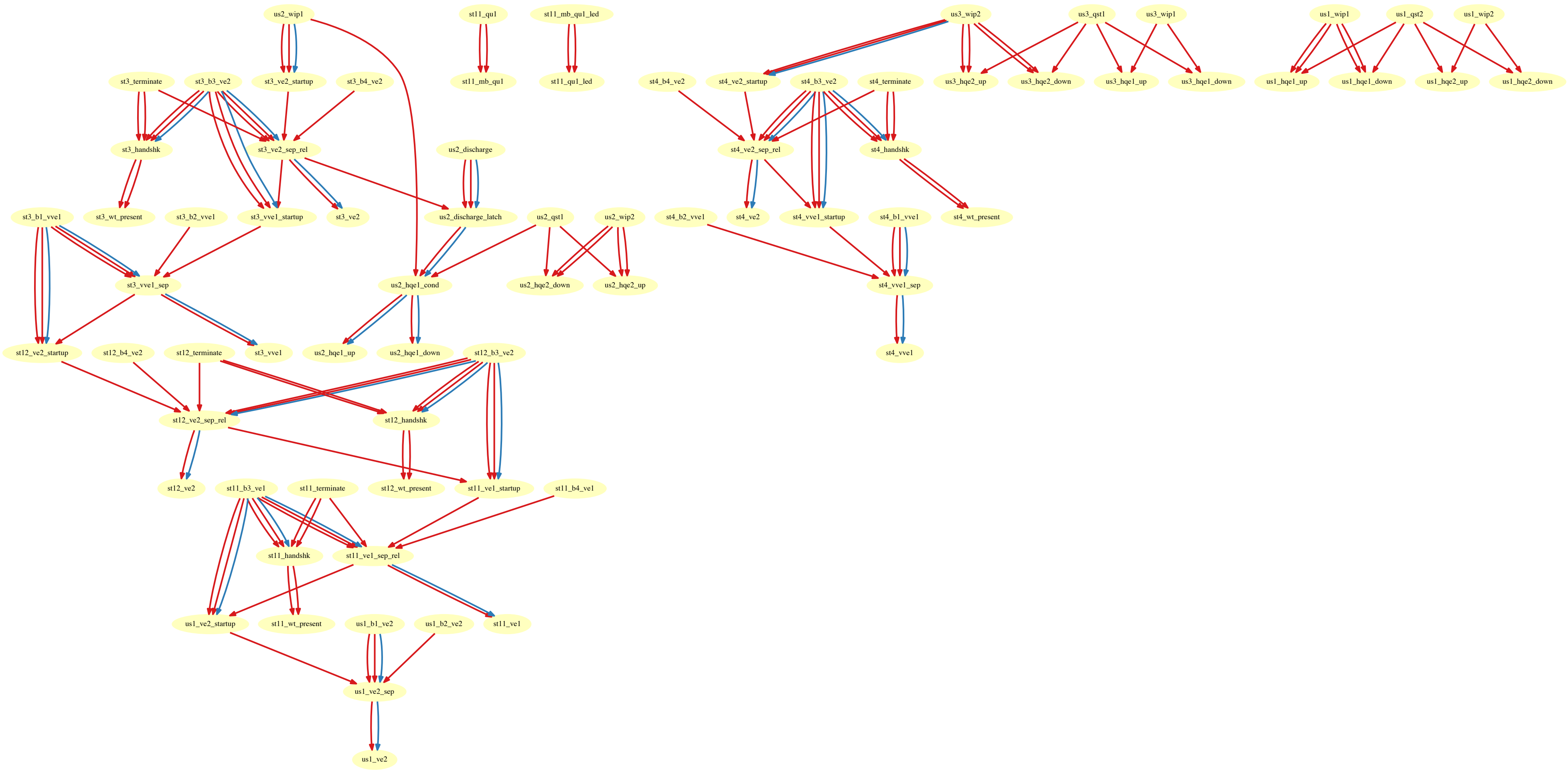
DIPL. PHYS. PEER STRITZINGER GMBH

Process connections



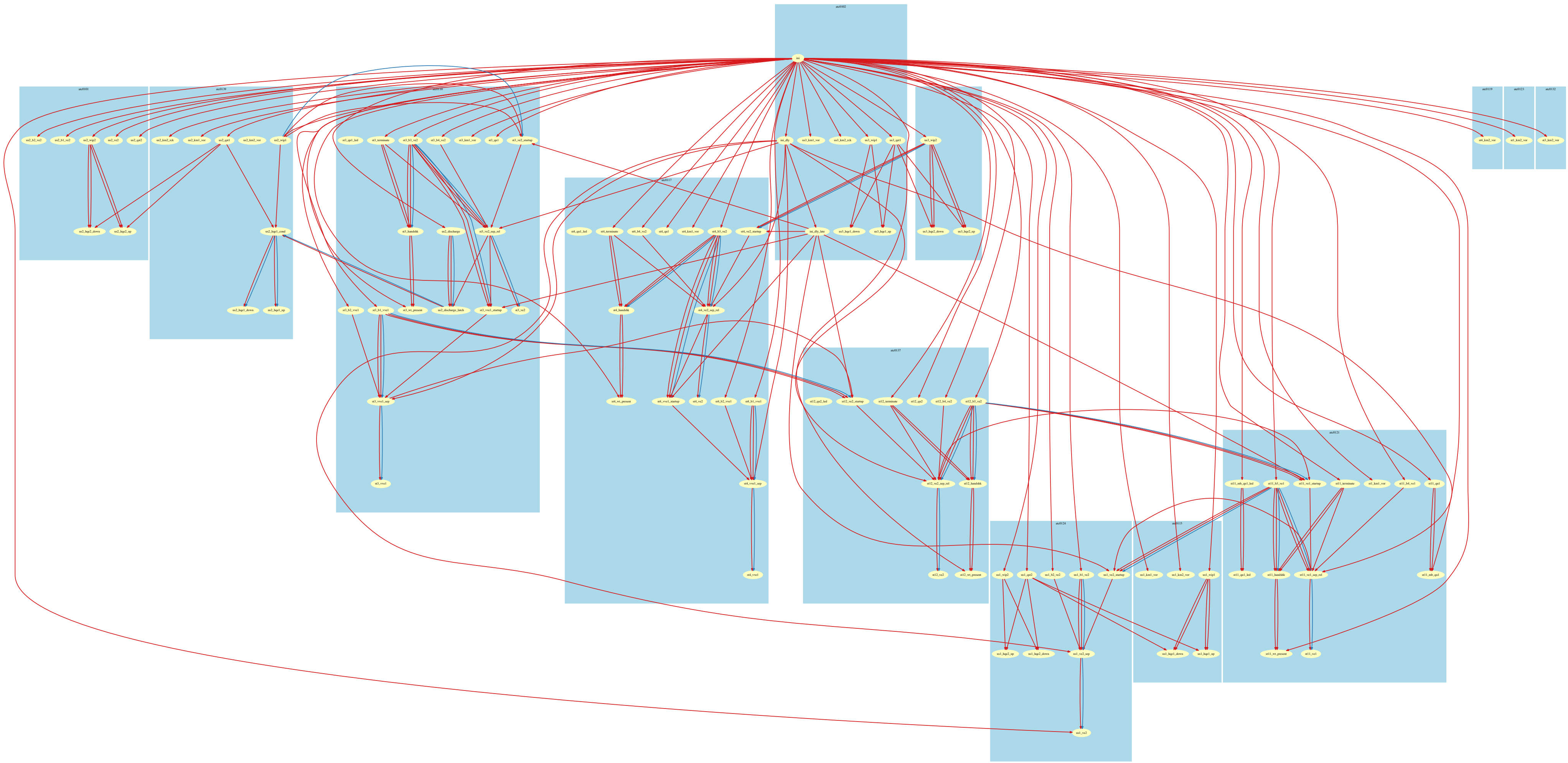
DIPL. PHYS. PEER STRITZINGER GMBH

High fanout removed



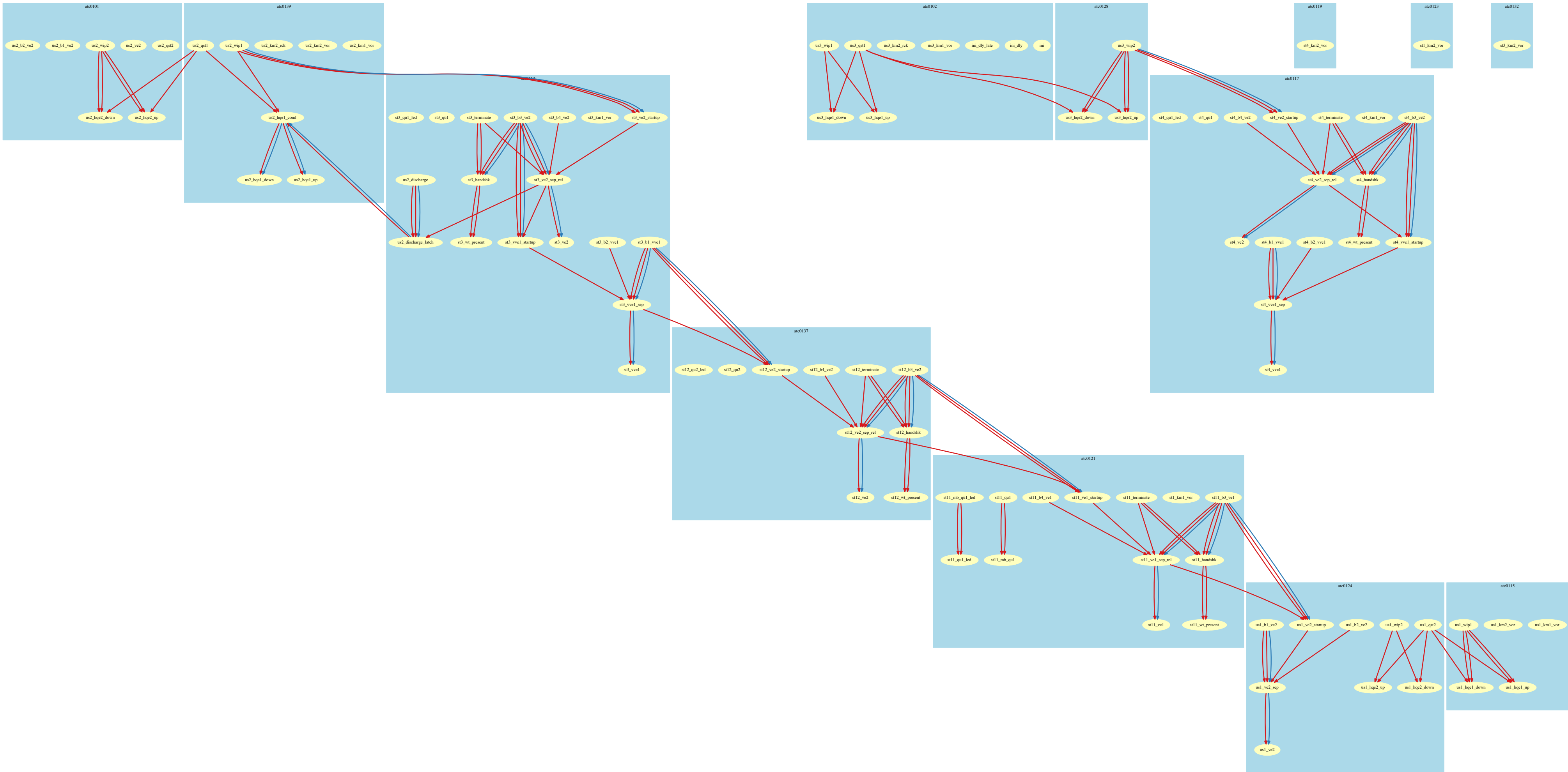
DIPL. PHYS. PEER STRITZINGER GMBH

Processes and Nodes



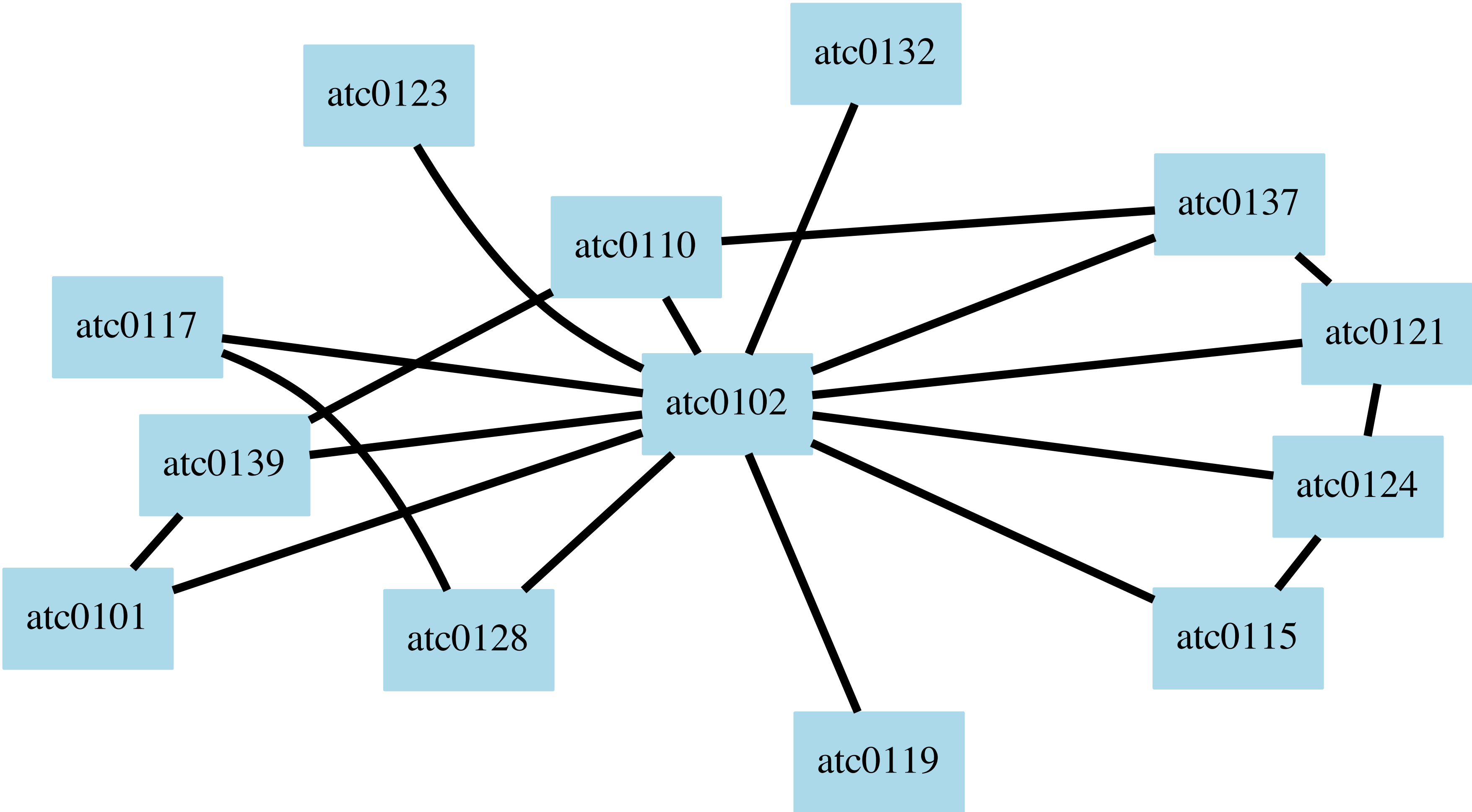
DIPL. PHYS. PEER STRITZINGER GMBH

High fanout removed

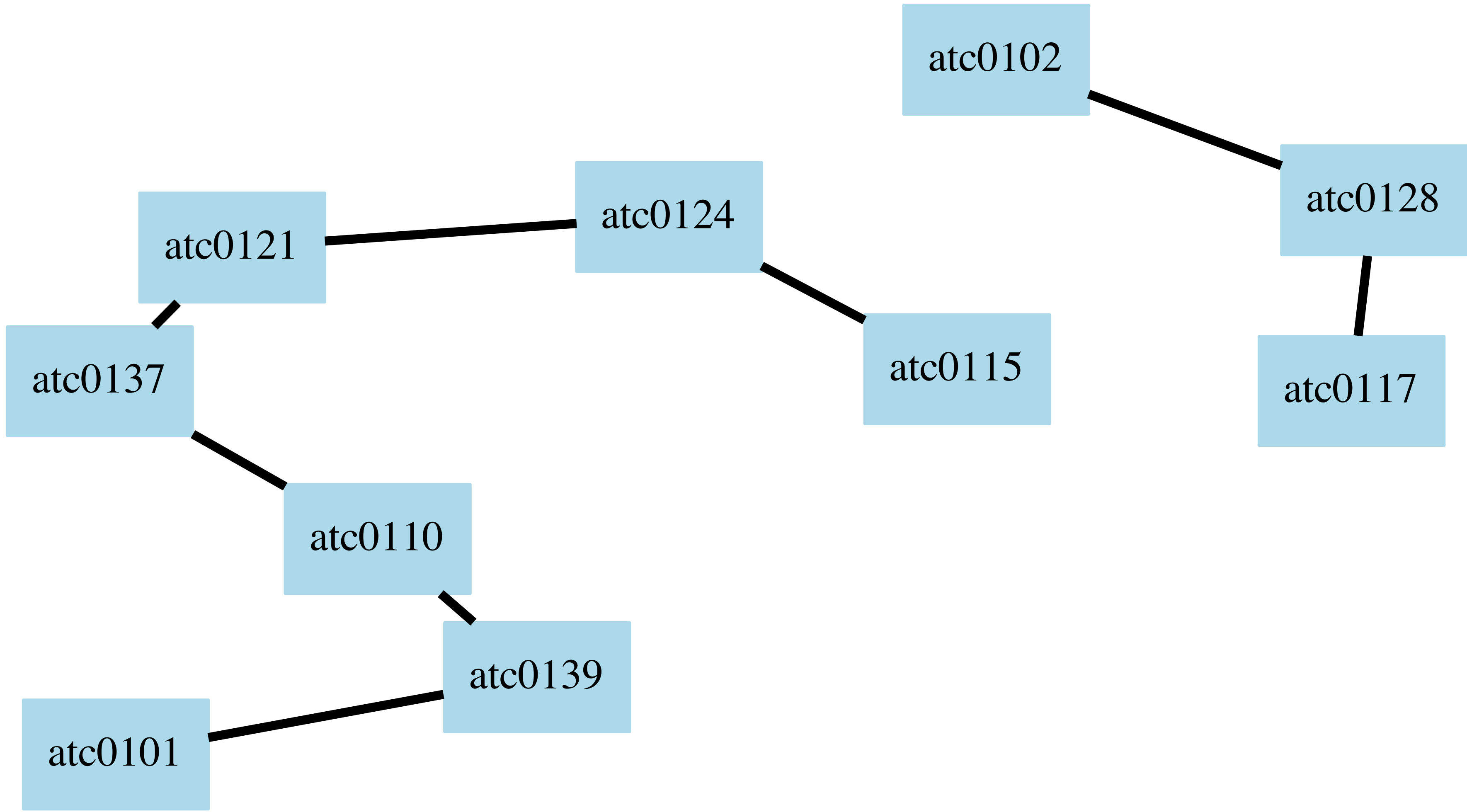


DIPL. PHYS. PEER STRITZINGER GMBH

Node Connections



High fanout removed



Fixing Erlang Distribution

- Fully connected network can't scale
- Head of line blocking of large messages
- Possible concurrency issues
- Hard Realtime networking
- Security in hostile networks

Plug in another distribution transport protocol

- Command line **-proto_dist** *mod* makes distribution call *mod_dist*
- Module implementing: childspecs/0, listen/1, accept/1, accept_connection/5, setup/4, close/1, select/1, is_node_name/1
- Port driver also needed, is called by erts_schedule_dist_command() see otp/erts/emulator/beam/dist.h
- More details http://www.erlang.org/doc/apps/erts/alt_dist.html

Invariants to be observed

- Any message from a node will be after nodeup and before nodedown
- Messages between any process pair need to stay ordered

epmd

- Some epmd support or replacement is possibly needed for other protocols
- Pure Erlang epmd implementation eases extending
- Already implemented to ease small embedded systems integration
- Open source https://bitbucket.org/peerst/epmd

Questions?

http://www.stritzinger.com
http://www.grisp.org
Twitter: @peerstr

peer@stritzinger.com

Thanks for funding the Erlang and RTEMS port IEC61499 compiler and being a great partner for innovations like this:

Rexroth
Bosch Group

The Drive & Control Company

DIPL. PHYS. PEER STRITZINGER GMBH