Specialeforsvar

”Implementering af NAT i XORP-projektet”

af Kristen Nielsen
kristen@diku.dk, krn@krn.dk

Datalogisk Institut ved Københavns Universitet.

Tirsdag den 25. marts 2008

Hvad går XORP-NAT-projektet ud på?

- Implementere NAT (netværks-adresse-oversættelse) i XORP-routerprojektet (eXtensible Open Router Project).
- Finde en mulig metode for at indbygge NAT-support i XORP.
- Lave et konfigurationssprog til konfiguration af NAT-delen i XORP-routeren.
- Implementere dele af løsningen.
- Evaluering af løsningen.

Resultatet af XORP-NAT-projektet:

- Generel model (arkitektur) for NAT-support i XORP er formuleret og besluttet.
- Model og terminologi for konfigurationssprog for XORP-NAT-modulet er formuleret og implementeret.
- Forslag til reduceret syntaks til specifikation af IP-adressedefinitioner, protokoller og porte.
- Design af XORP-NAT-modulet.
- Implementering af det meste af XORP-NAT-modulet. (POC)
- Evalueret løsningen.

Program

- Præsentation af specialet (45 min)
  - Del 1. Indledning
  - Del 2. Detaljeret gennemgang af udvalgte dele af projektet
  - Del 3. Evaluering og konklusion
- Kort pause
- Spørgsmål

Program

- Præsentation af specialet (45 min)
  - Del 1. Indledning

    Kort introduktion til:
    - XORP-projektet og NAT-implementationen
    - Realms,
    - NAT vs. NAPT, basal NAT-funktion,
    - IP-adresser, protokoller, porte, offentlige vs. private IP-adresser.
  - Del 2. Detaljeret gennemgang af udvalgte dele af projektet
  - Del 3. Evaluering og konklusion
- Kort pause
- Spørgsmål

Program

- Præsentation af specialet (45 min)
  - Del 1. Indledning
  - Del 2. Detaljeret gennemgang af udvalgte dele af projektet
    - Udvidelsen af XORPs konfigurationssprog med support for NAT
    - Kort om implementationen af XORP-NAT-modulet
  - Del 3. Evaluering og konklusion
- Kort pause
- Spørgsmål

Program

- Præsentation af specialet (45 min)
  - Del 1. Indledning
  - Del 2. Detaljeret gennemgang af udvalgte dele af projektet
  - Del 3. Evaluering og konklusion
    - Mine bidrag
    - De valgte løsninger
    - Implementeringen
    - Ydelsesaspekter
    - Konklusionen
- Kort pause
- Spørgsmål

Program

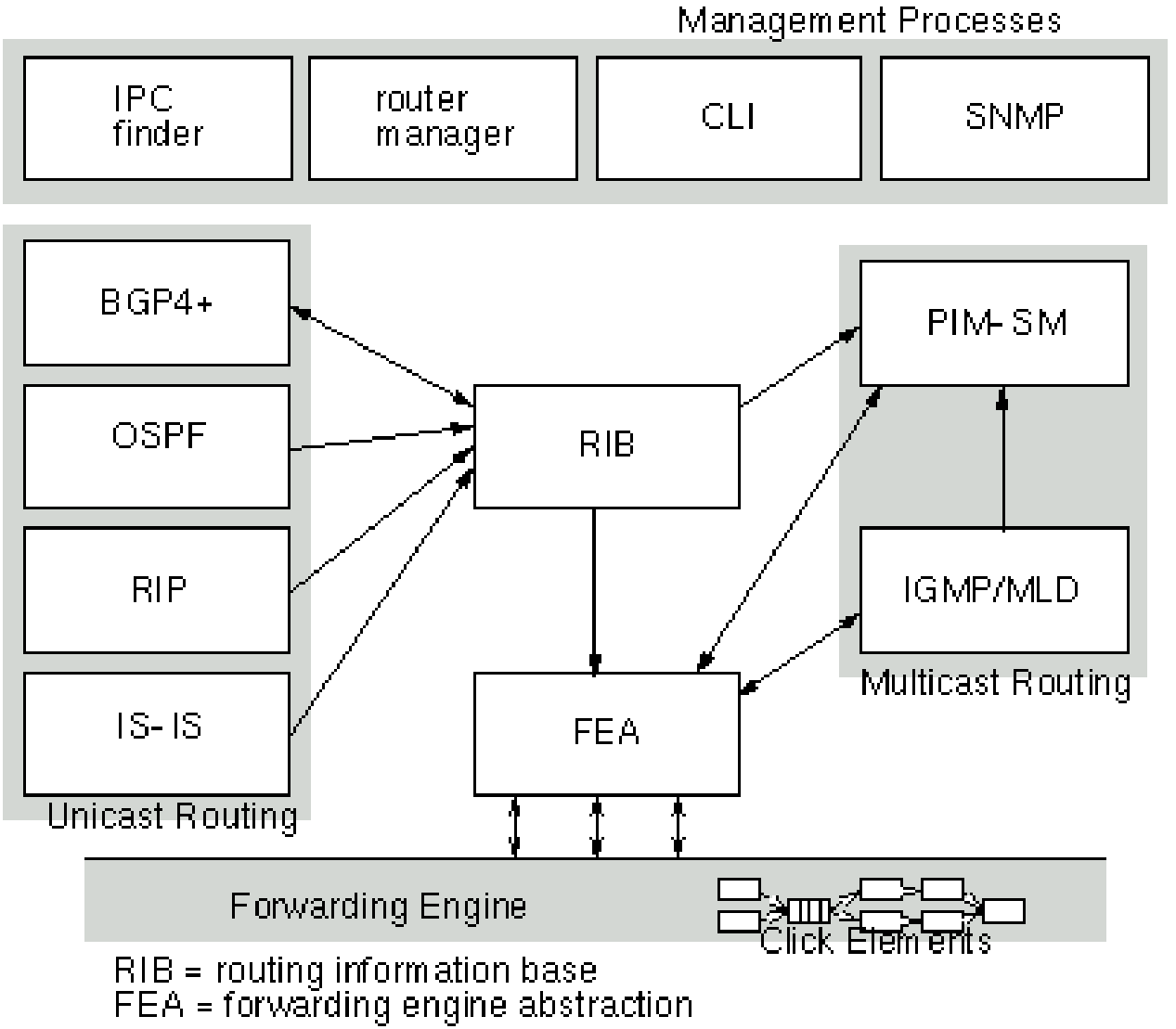
- Præsentation af specialet (45 min)
  - **Del 1. Indledning**

    Kort introduktion til:

    - **XORP-projektet og NAT-implementationen**
    - **Realms,**
    - **NAT vs. NAPT, basal NAT-funktion,**
    - **IP-adresser, protokoller, porte, offentlige vs. private IP-adresser.**
  - Del 2. Detaljeret gennemgang af udvalgte dele af projektet
  - Del 3. Evaluering og konklusion
- Kort pause
- Spørgsmål

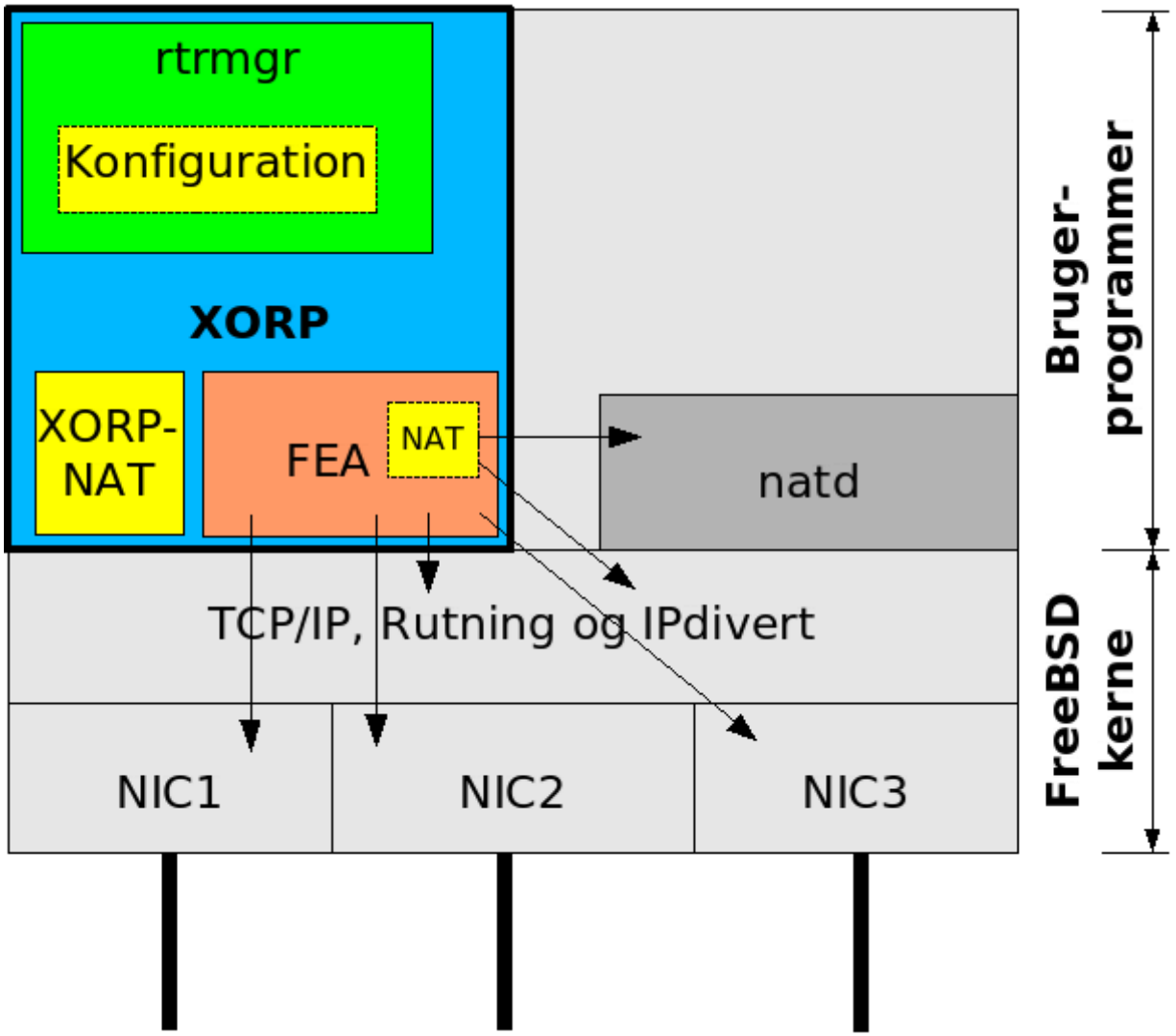
Introduktion: Overordnet arkitektur for XORP-projektet

- XORP-processer taler sammen via XRLs
- Multiprocess og enkelt trådet arkitektur
- FEA er interfacet imod IP-laget. Resten er O.S. uafhængig kode



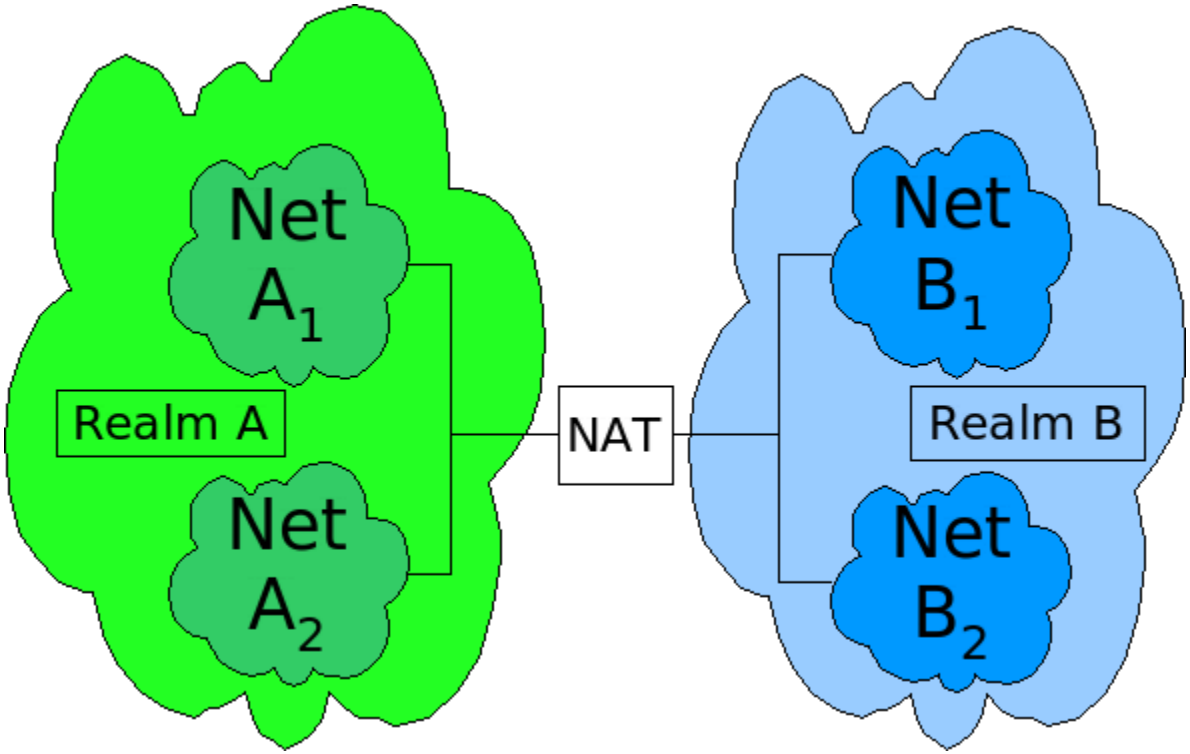
Introduktion: overordnet arkitektur for XORP-NAT-projektet

- XORP afvikles som brugerapplikation.
- FreeBSD natd-programmet er selve NAT-funktionen.
- FreeBSDs IPdivert anvendes til at dirigere trafik til/fra kernen og natd.



Introduktion: Realms

- NAT-enheder oversætter IP-adresser imellem 2 IP-realms.
- Kun 1 realm på hver side af NAT-enheden.
- Flere IP-net pr. realm.

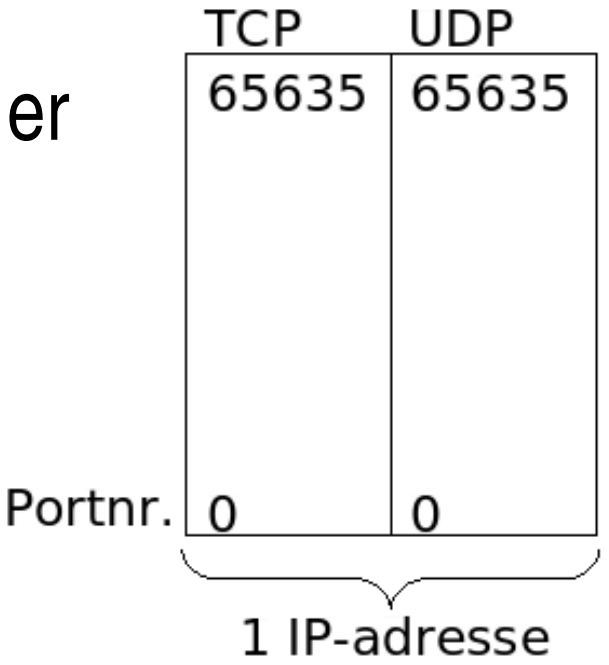


Intro: IP-adresser (offentlige / private)

- Offentlige IP-(unicast)adresser, som må anvendes på Internettet.
    - 0.0.0.0 – 223.255.255.255 (undtaget de nævnte private IP-adresser)
- Private IP-(unicast)adresser, kan anvendes mange steder, men aldrig på Internettet.
    - 10.0.0.0 – 10.255.255.255
    - 172.16.0.0 – 172.31.255.255
    - 192.168.0.0 – 192.168.255.255

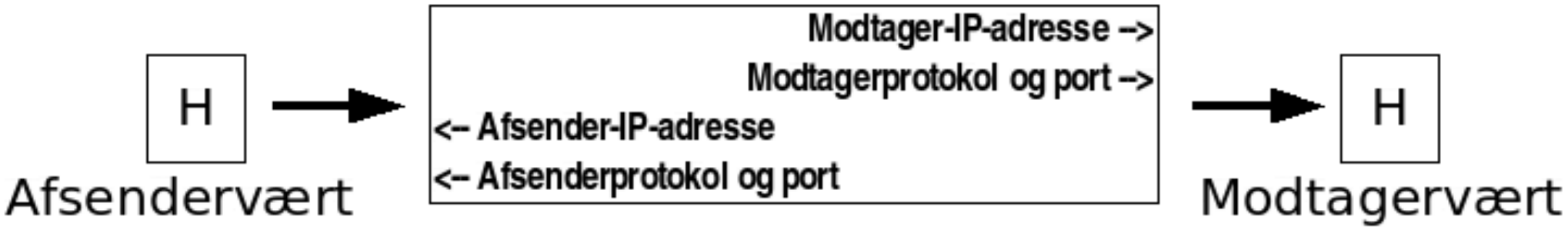
Introduktion: TCP, UDP og portadresser

- Hver IP-adresse har 2 typer serviceadresser der kaldes porte.
  Disse kaldes TCP- og UDP-porte og der er 65.536 af hver slags.
  (fra 0 til 65535)
  - TCP: Transmissions-kontrol-protokol
  - UDP: User-datagram-protocol



Introduktion: IP-pakker

- IP-pakker består hovedsageligt af:
  - Afsenderadresse, protokol og portnummer
  - Modtageradresse, protokol og portnummer
  - Data som sendes til modtageren



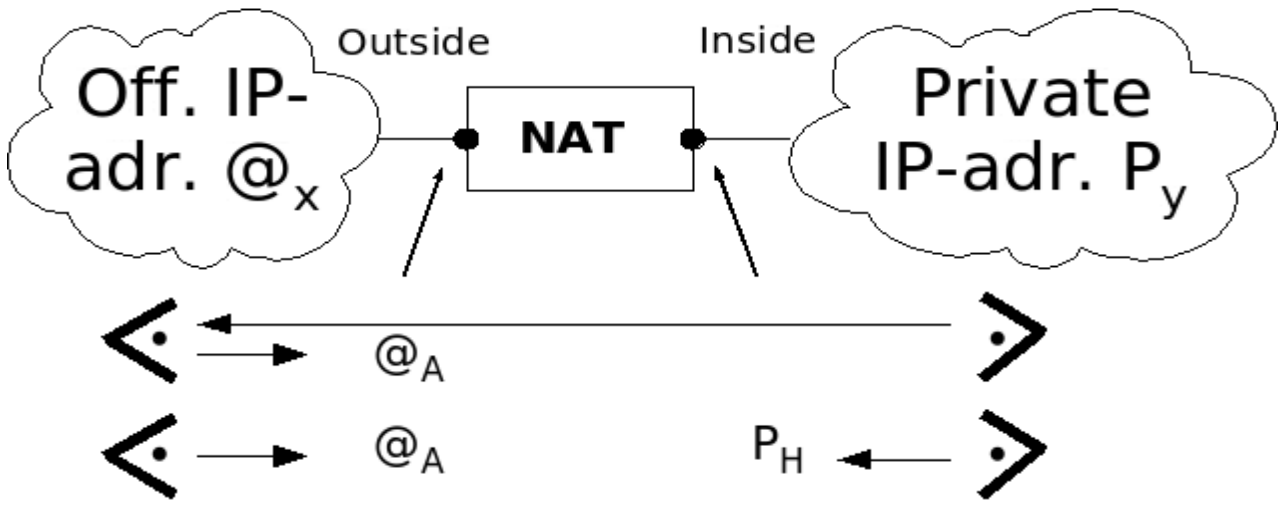
Introduktion: NAT vs. NAPT (netværksadresse- og portoversættelse)

- De fleste som kender til NAT har mødt dette i forbindelse med deling af en Internetforbindelse imellem flere computere.
- Der findes 2 grundlæggende typer: NAT og NAPT.
- I daglig tale siger vi NAT men vi mener (og anvender) oftest NAPT.
  - NAT oversætter kun IP-adresser
  - NAPT oversætter både IP-adresser og TCP/UDP-portnumre

Introduktion: 3 typer NAT/NAPT.

- Statisk-NAT.
  - Oversætter adresser udfra en (forud)konfigureret tabel. Anvendes oftest ved oversættelse af IP-strømme oprindende fra ydersiden af NAT-funktionen. Kaldes også 1:1 oversættelse.
- Dynamisk-NAT.
  - Opretter og nedlægger automatisk adresseoversættelsesregler udfra regler i konfigurationen. Anvendes hovedsageligt ved oversættelse af IP-strømme oprindende fra indersiden af NAT-funktionen
- Loadsharing-NAT (LS-NAT).
  - Fordeler henvendelser til en fælles serviceadresse på ydersiden imellem en række foruddefinerede servere placeret på indersiden.

Introduktion: NAT basal funktion.



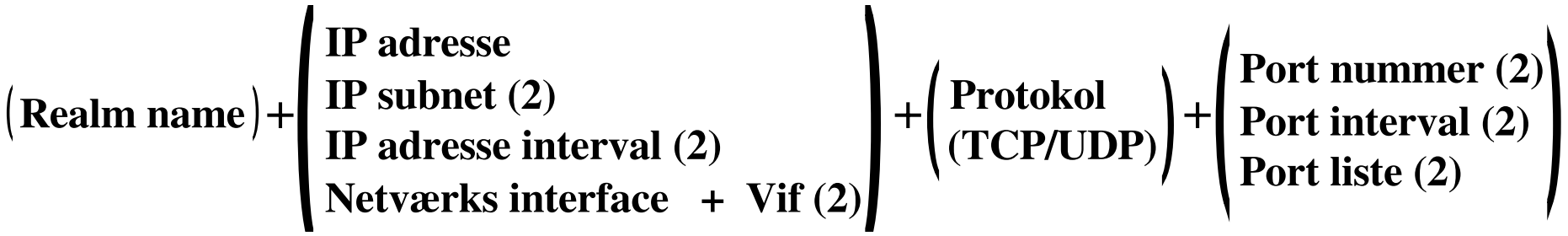
- NAT-modulet oversætter P_y afsenderadressen til $@_A$ udgående, og modtageradressen $@_A$ til P_y indgående.
- Et særligt tilfælde er når NAT-modulet også oversætter $@_X$ værtens IP-adresse til P_H i det private IP-område.
  - Dvs. både modtager- og afsenderadresser oversættes. IP-verdenen på den modsatte side af NAT-modulet skjules dermed helt.

Program

- Præsentation af specialet (45 min)
  - Del 1. Indledning
  - **Del 2. Detaljeret gennemgang af udvalgte dele af projektet**
    - **Udvidelsen af XORPs konfigurationssprog med support for NAT**
    - **Kort om implementationen af XORP-NAT-modulet**
  - Del 3. Evaluering og konklusion
- Kort pause
- Spørgsmål

XORP-NAT-konfigurationssprog 1/7

- IP-definitioner for inside og outside siden af NAT-modulet består af 4 dele: Realm, IP, protokol og porte med flere varianter for nogle af parametrene:



- Der kan være 1 eller flere af sådanne definitioner på hver side af NAT-modulet, alle definitioner, på en side, skal tilhøre samme realm.

XORP-NAT-konfigurationssprog 2/7
Den implementerede version

nat {

static-nat { **map** <id> { **inside** <id> {<ip-def>} **outside** <id>{<ip-def>} }}

dynamic-nat { **map** <id> { **inside** <id> {<ip-def>} **outside** <id>{<ip-def>}}}

ls-nat { map <id> { **inside** <id> {<ip-def>} **outside** <id>{<ip-def>}}}

}

XORP-NAT-konfigurationssprog 3/7
Den implementerede version

inside <id> { /* **outside** har samme format */

realm: <realm:txt>

ip-address: <ip:ipv4>

ip-range: <ipfrom:ipv4> - <ipto:ipv4>

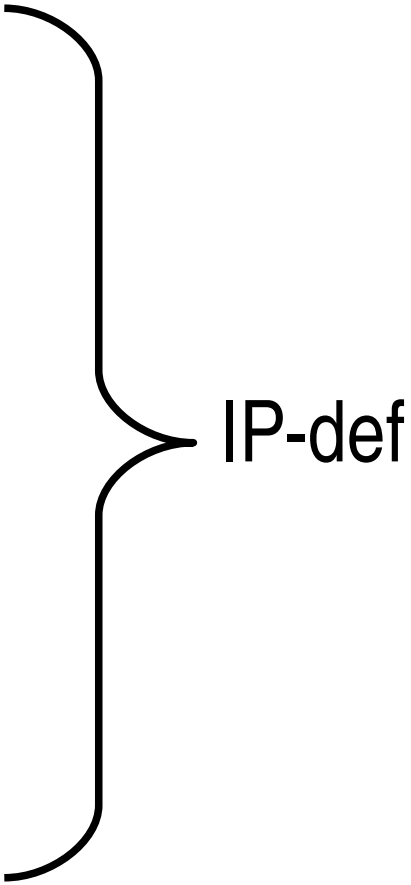
ipnet: <ipnet:ipv4net>

interface: <interface:txt>

vif: <vif:txt>

protocols: <protocols:txt>

}



XORP-NAT-konfigurationssprog 4/7
Den implementerede version

protocols: <protocols.txt>

- Argumentet er et sprog der beskriver protokoller og porte (sub-protokoller).
- Eks: **protocols**: tcp: 22-23, 80, udp: 53
- Ovenstående specificerer tcp protokol port 22,23 og 80 og udp protokol port 53

XORP-NAT-konfigurationssprog 5/7
Forslag til optimeret IP-def syntaks

- IP-adresse: 10.10.10.10, IP-net: 10.10.10.0/24,
- IP-interval: 10.10.10.10 .. 10.10.10.15
- Protokol og port: tcp: 20-23, 80
- IP, protokol og porte: 10.10.10.0/24: tcp: 20-23, 80
- eksempel 1

inside <id> {

realm: <id>

ip: <IP-def>

}

- eksempel 2:

inside <id> {

ip: realm: <realm-id> ip: <IP-def>

}

XORP-NAT-konfigurationssprog 6/7
Forslag til reduceret konfigurationssprog

Implementerede version

```
 /* inside og outside har samme format */
inside homenet {
    realm: inside
    ip-address: 10.10.10.10
    ip-range:
    ipnet:
    interface:
    vif:
    protocols: tcp: 22
}
```

Foreslåede ny version

```
 /* inside og outside har samme format */
inside homenet {
    realm: inside
    ip: 10.10.10.10: tcp: 22
}
```

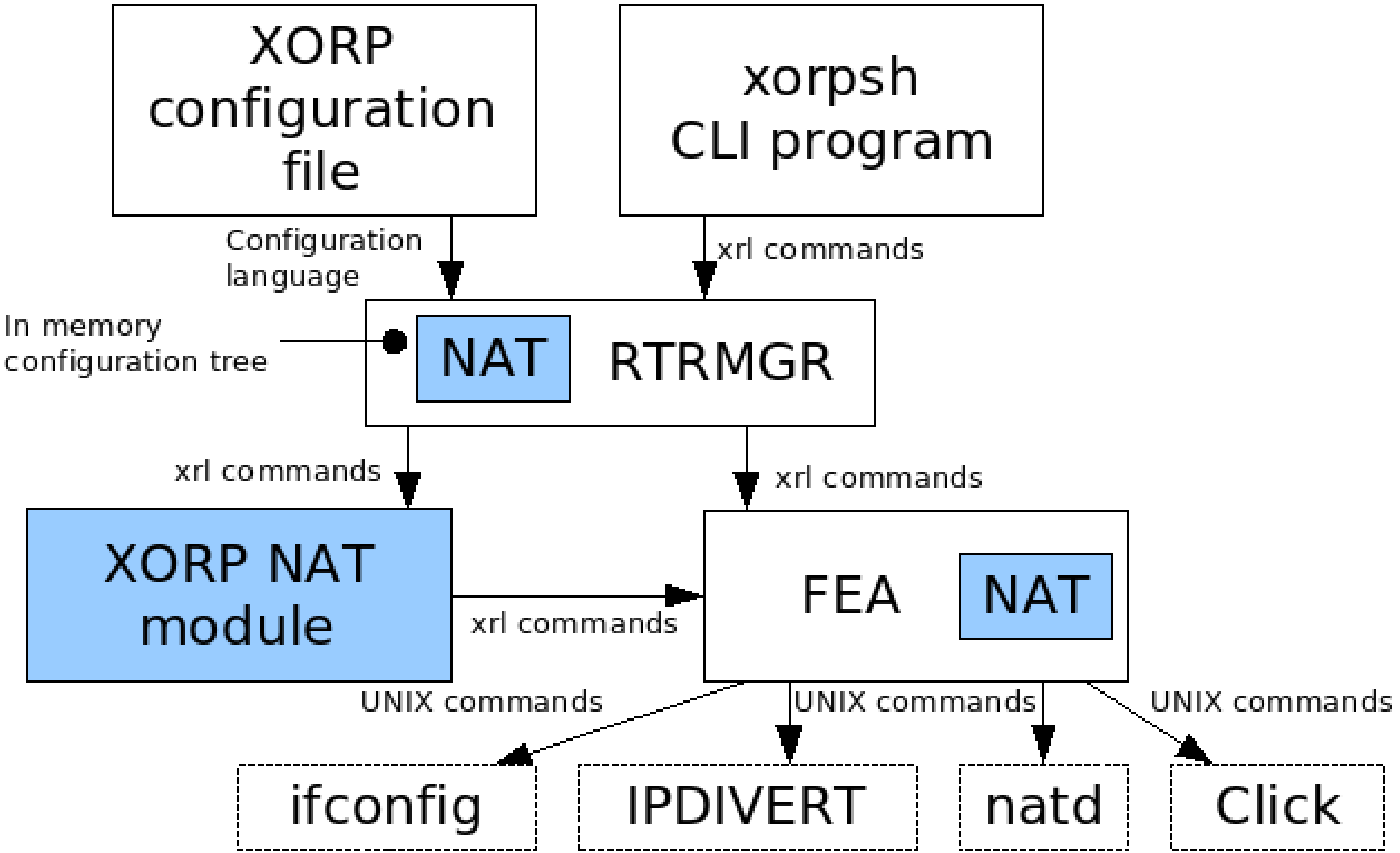
XORP-NAT-konfigurationssprog 7/7 Opsummering

- Sproget er designet så det passer til alle NAT-moduler.
- Det store antal konfigurationsparametre på hver side af NAT-modulet bør reduceres.
- Der er foreslået en reduceret syntaks der gør det muligt at udtrykke tilsvarende rige konfigurationer på kun få linier.
- NAT-tabel-timeouts er ikke suporteret i sproget.

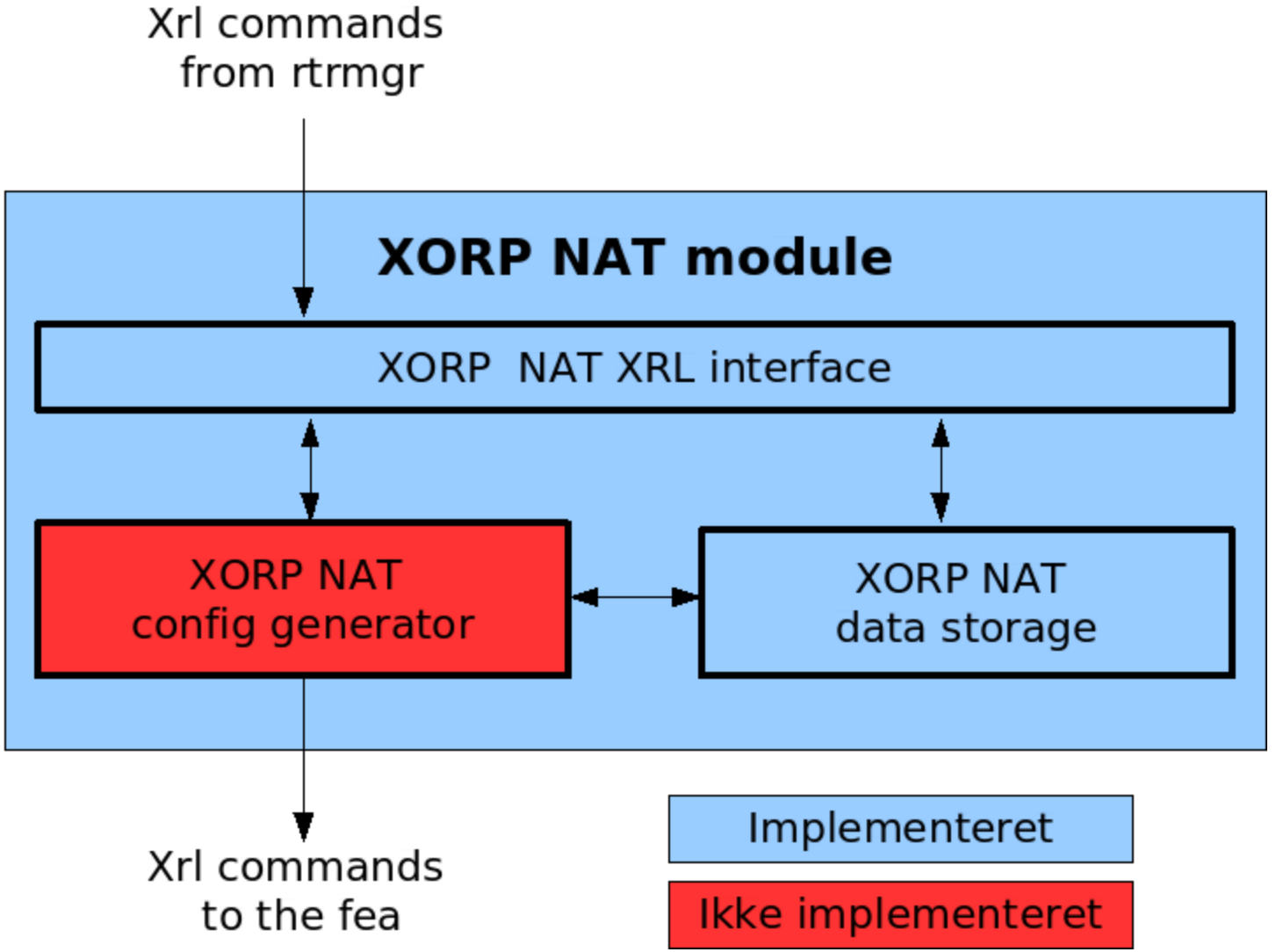
Implementering af XORP-NAT-modulet 1/4

- NAT modulet er en selvstændig process under XORP.
- Skrevet i C++ (som resten af XORP).
- Templates til rtrmgr og xorpsh, og xrl interfaces lavet.
- Undervejs var jeg forbi en række tools: gnu autoconf og automake som skulle kende til NAT-delen af XORP.

Implementering af XORP-NAT-modulet 2/4 - overblikstegning - kommandoflow



Implementering af XORP-NAT-modulet 3/4



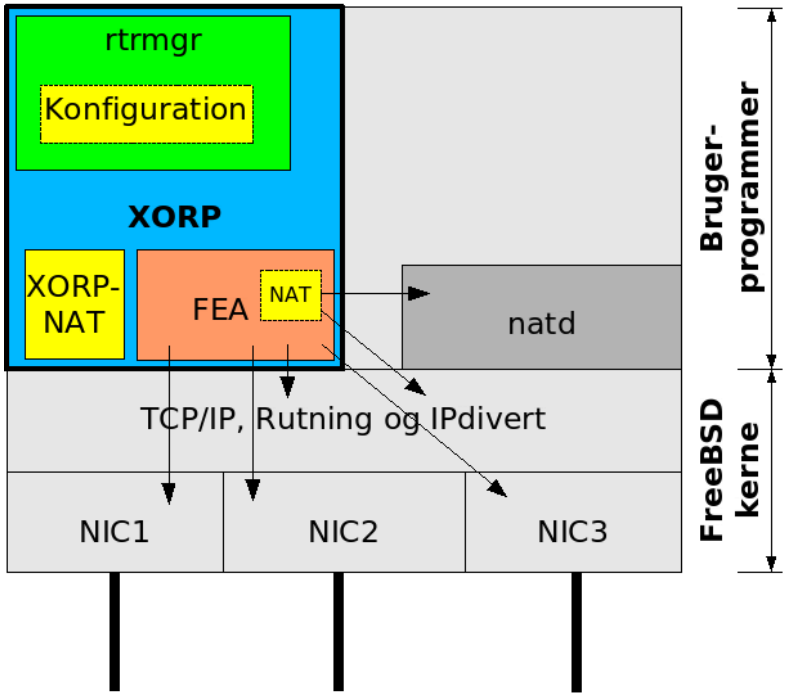
Implementering af XORP-NAT 4/4 Opsummering

- XRL og datastoragedelen er implementeret og testet.
- Alt er integreret med XORPs autoconfig og build system.
- XORP-NAT-modulet er testet selvstændigt, kun testet lidt sammen med resten af XORP-systemet.

Program

- Præsentation af specialet (45 min)
  - Del 1. Indledning
  - Del 2. Detaljeret gennemgang af udvalgte dele af projektet
  - **Del 3. Evaluering og konklusion**
    - **Mine bidrag**
    - **De valgte løsninger**
    - **Implementeringen**
    - **Ydelsesaspekter**
    - **Konklusionen**
- Kort pause
- Spørgsmål

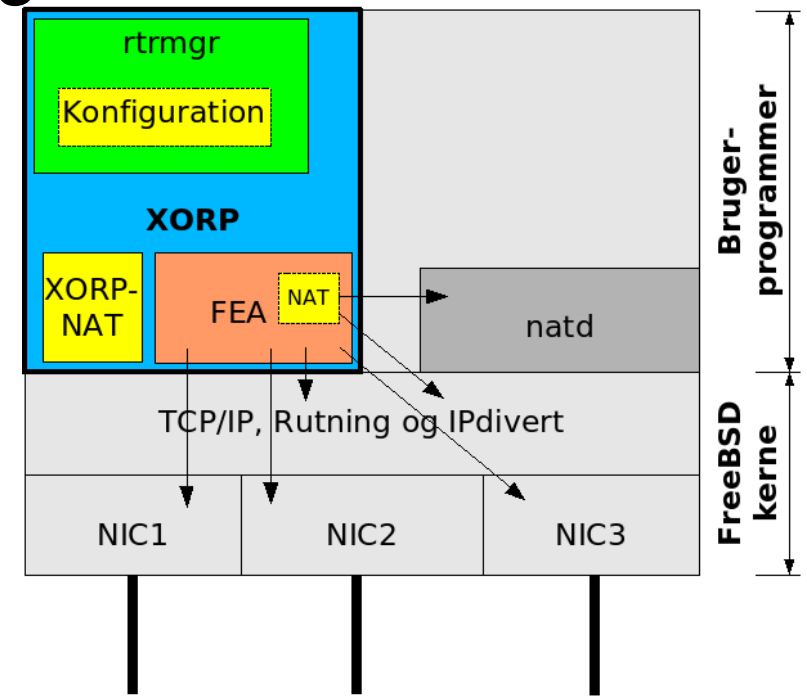
Evaluering og konklusion 1/6
Mine bidrag



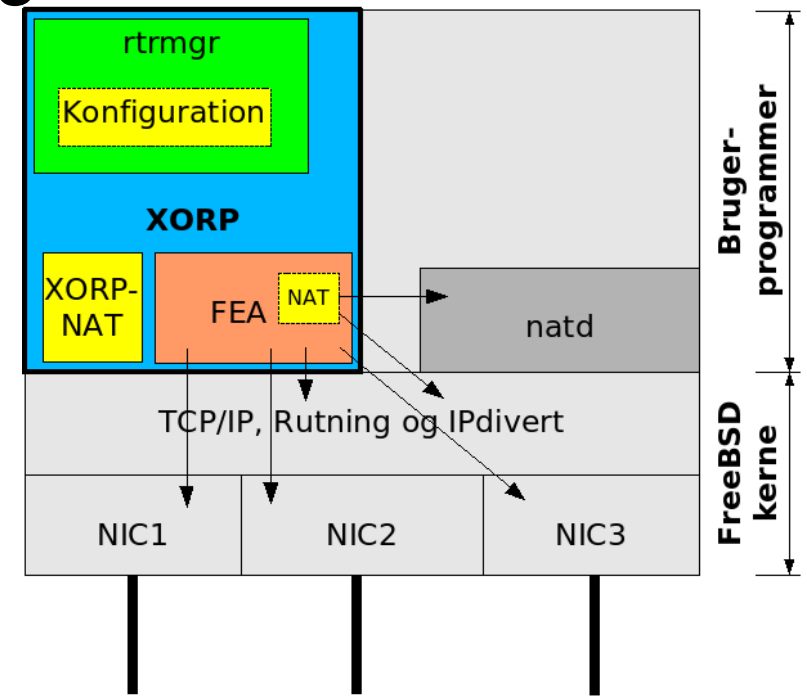
- Analyse af opgaven og XORP
- Fundet en generel løsning der rummer samspillet med fremtidige NAT-moduler. (f.eks. fra nye O.S.)
- Implementeret løsningen til et vist niveau, men før det bliver open-source-software med god kvalitet skal det færdiggøres, pudses af og testes grundigere.

Evaluering og konklusion 2/6
De valgte løsninger

- Generelle løsninger er søgt i fremfor for ad-hoc løsninger overalt hvor dette var muligt.
- NAT-modulet implementeret som et selvstændigt XORP-modul.
- Konfigurationen af NAT-modulet er løst med et mix af eksisterende XORP-konfigurationselementer og ved tilføjelse af et sprog for protokoller og porte. Forslag til udvidelse af dette til at dække hele IP-delen af NAT-konfigurationen.
- Generelt udvidbar kode der muliggør udvidelse f.eks. med nye protokoltyper og med henblik på genanvendelse andre steder i XORP hvis projektet vælger dette.

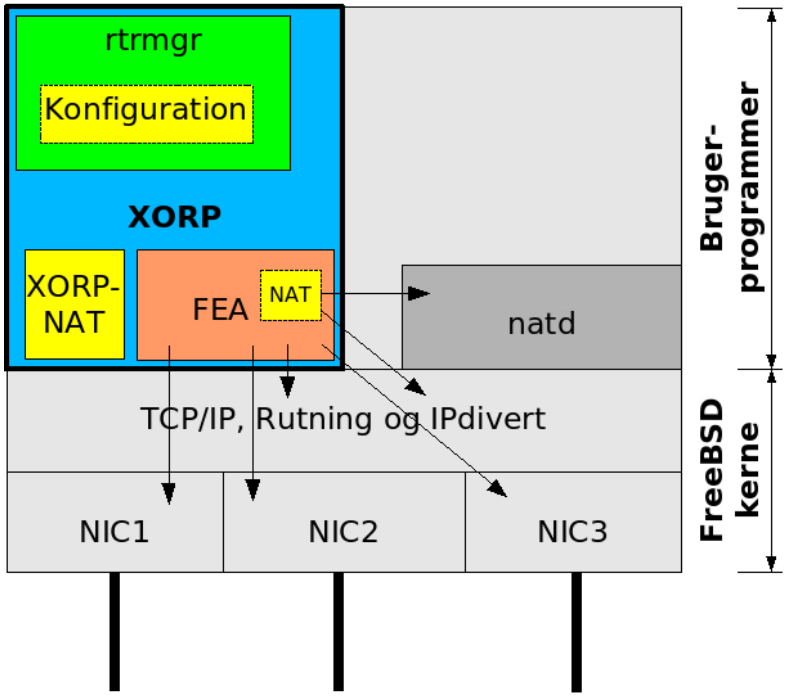


Evaluering og konklusion 3/6
De valgte løsninger



- NAT-datastorageklassen komprimerer de lagrede port-definitioner, således at disse altid udlæses i deres mest kompakte form.
- Realms er indført fordi jeg forventer at XORP/FreeBSD snart får kerne support for virtuelle IP-stakke. Indtil da kan realms tildeles default værdier som fastsættes i templaten. F.eks. ”inside” og ”outside” for et NAT-modul.

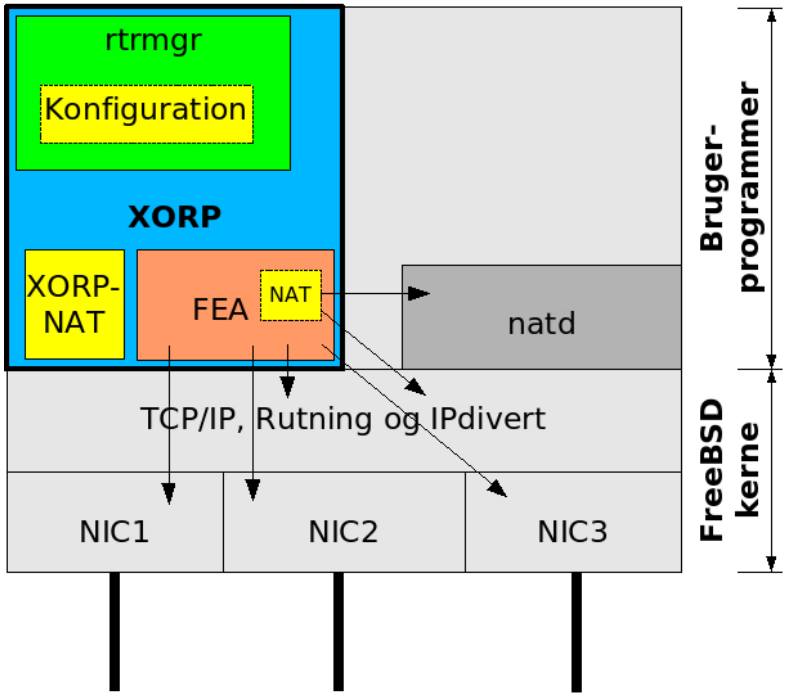
Evaluering og konklusion 4/6
Implementeringen



- Hensigten at ændre færrest steder i den eksisterende XORP-kode-base, og have minimale afhængigheder til anden XORP-kode.
- Tilføjelse af ny kode fremfor at tilrette eksisterende kode.
- Anvende eksisterende templates og værktøjer i XORP, hvor dette er muligt.
- Muligør let decentral vedligeholelse af koden.

Evaluering og konklusion 5/6
Ydelsesaspekter

- Hvilke komponenter påvirker XORP-systemets ydelse?
  - Ydelse af ruteadministration og systemkonfiguration
  - Ydelsen af IP-pakkeflow (Rutning, IP-divert og NAT)



- XORPs effektivitet har kun betydningfor hvor hurtigt en ændring i systemet formidles ud til IP-forwarding delen.
- Værtens Netværkskort, TCP-stak og NAT-modul er de eneste elementer der har indflydelse på systemets ydelse rent rutnings- og NAT-mæssigt.
- Effektiviteten afhænger derfor udelukkende af værtssystemets ydelse.

Evaluering og konklusion 6/6
Konklusionen

- Projektet har været en succes.
  - Der er fundet en god generel løsning
  - Den opfylder de stillede krav.
  - Der er del-løsninger der rækker videre end vores brug, hvis projektet ønsker det.
- Punkter hvor jeg er mindre tilfreds
  - Syntaksen: 10.10.10.0/28: tcp: 22 kunne være pænere. (1. kolon)
  - Syntaksen: tcp: 22-23, 80, udp: 53 (2. komma)
  - Termen ”sub-protokol” er det bedste bud jeg kunne finde, men det er ikke helt godt. Heldigvis indgår den ikke direkte i brugerdialoger.

Program

- Præsentation af specialet (45 min)
  - Del 1. Indledning
  - Del 2. Detaljeret gennemgang af udvalgte dele af projektet
  - Del 3. Evaluering og konklusion
- Kort pause
- Spørgsmål

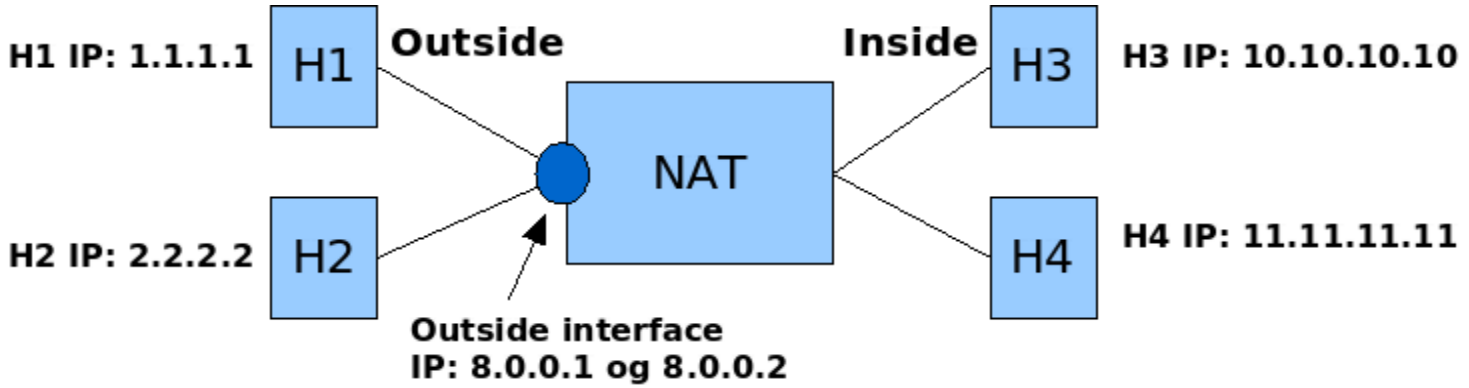
Program

- Præsentation af specialet (45 min)
  - Del 1. Indledning
  - Del 2. Detaljeret gennemgang af udvalgte dele af projektet
  - Del 3. Evaluering og konklusion
- Kort pause
- Spørgsmål

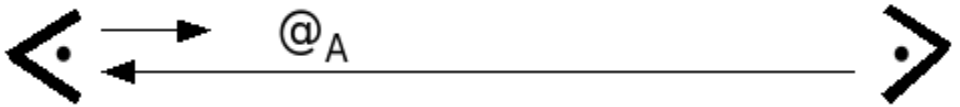
Spørgsmål

The END
(og forfriskninger)

Statisk-NAT (ikke NAPT)



Entry	Outside	Inside
static 1	8.0.0.1	10.10.10.10
static 2	8.0.0.2	11.11.11.11



TCP H1 -> H3 (Outside interface)

Src-IP:	1.1.1.1
Dest-IP:	8.0.0.1
Src-Port:	40000
Dest-Port:	80

TCP H1 -> H3 (Inside interface)

Src-IP:	1.1.1.1
Dest-IP:	10.10.10.10.
Src-Port:	40000
Dest-Port:	80

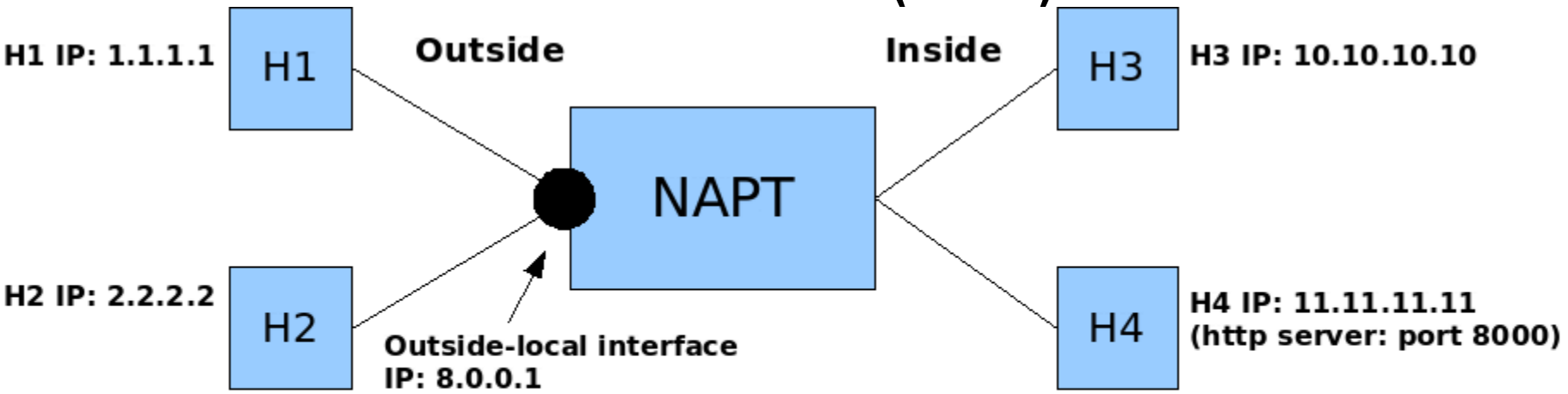
TCP H1 <- H3 (Outside interface)

Src-IP:	8.0.0.1
Dest-IP:	1.1.1.1
Src-Port:	80
Dest-Port:	40000

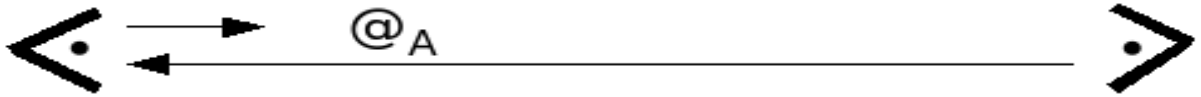
TCP H1 <- H3 (Inside interface)

Src-IP:	10.10.10.10
Dest-IP:	1.1.1.1
Src-Port:	80
Dest-Port:	40000

Statisk-NAPT (kort)



Entry	Outside-source	Outside-local	Inside-local
static-3	-	8.0.0.1:80	11.11.11.11:8000



TCP H1 -> H4 (Outside interface)

Src-IP:	1.1.1.1
Dest-IP:	8.0.0.1
Src-Port:	30000
Dest-Port:	80

TCP H1 -> H4 (Inside interface)

Src-IP:	1.1.1.1
Dest-IP:	11.11.11.11
Src-Port:	30000
Dest-Port:	8000

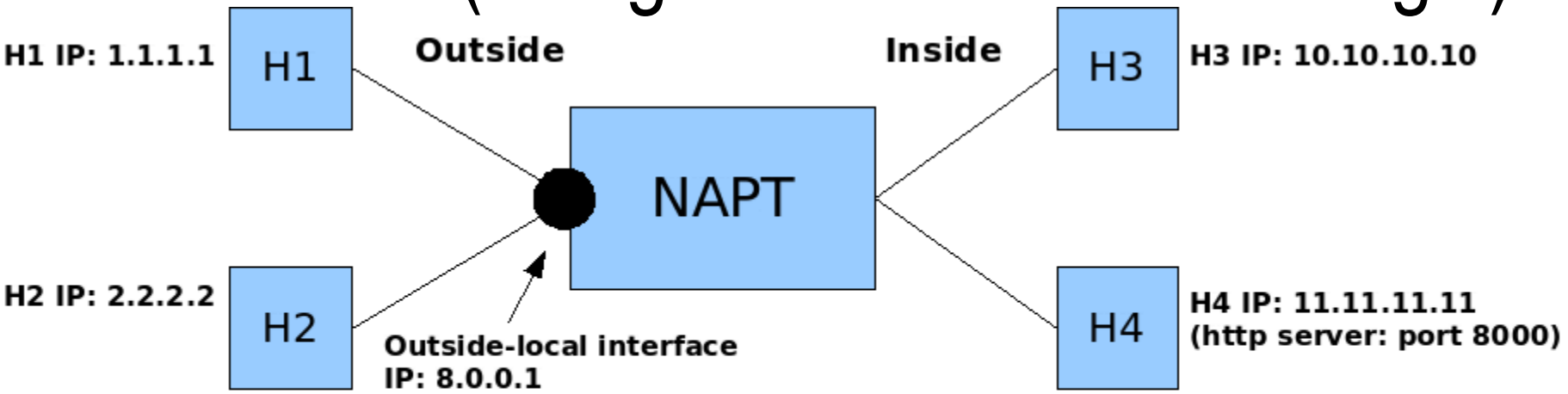
TCP H1 <- H4 (Outside interface)

Src-IP:	8.0.0.1
Dest-IP:	1.1.1.1
Src-Port:	80
Dest-Port:	30000

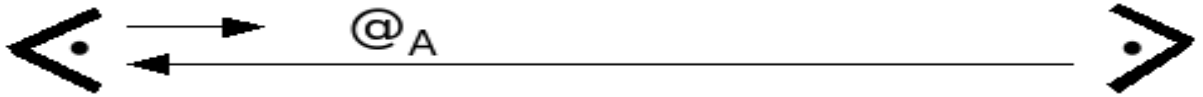
TCP H1 <- H4 (Inside interface)

Src-IP:	11.11.11.11
Dest-IP:	1.1.1.1
Src-Port:	8000
Dest-Port:	30000

Statisk-NAPT (manglende oversættelsesregel)



Entry	Outside-source	Outside-local	Inside-local
static-3	-	8.0.0.1:80	11.11.11.11:8000



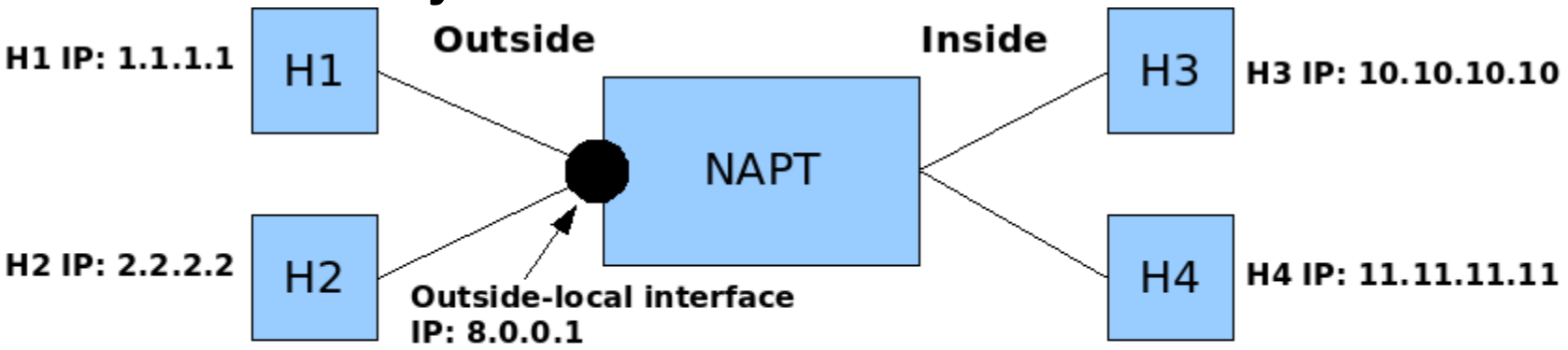
TCP H1 -> H3 (Outside interface)

Src-IP:	1.1.1.1
Dest-IP:	8.0.0.1
Src-Port:	40000
Dest-Port:	25

TCP H1 -> H3 (Inside interface)

Src-IP:	
Dest-IP:	
Src-Port:	
Dest-Port:	

Dynamisk-NAT/NAPT



Entry	Outside-source	Outside-local	Inside-local
dynamic-1	1.1.1.1:80	8.0.0.1:40000	10.10.10.10:40000

TCP H1 <- H3 (Outside interface)

Src-IP:	8.0.0.1
Dest-IP:	1.1.1.1
Src-Port:	40000
Dest-Port:	80

TCP H1 <- H3 (Inside interface)

Src-IP:	10.10.10.10
Dest-IP:	1.1.1.1
Src-Port:	40000
Dest-Port:	80

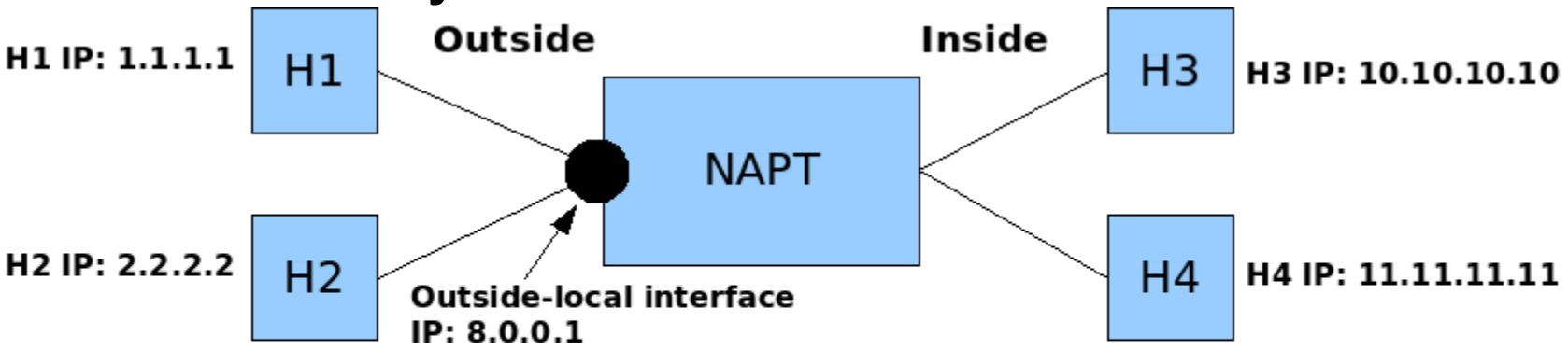
TCP H1 -> H3 (Outside interface)

Src-IP:	1.1.1.1
Dest-IP:	8.0.0.1
Src-Port:	80
Dest-Port:	4000

TCP H1 -> H3 (Inside interface)

Src-IP:	1.1.1.1
Dest-IP:	10.10.10.10
Src-Port:	80
Dest-Port:	40000

Dynamisk-NAT/NAPT



Entry	Outside-source	Outside-local	Inside-local
dynamic-1	1.1.1.1:80	8.0.0.1:40000	10.10.10.10:40000
dynamic-2	1.1.1.1:80	8.0.0.1:40001	11.11.11.11:40000

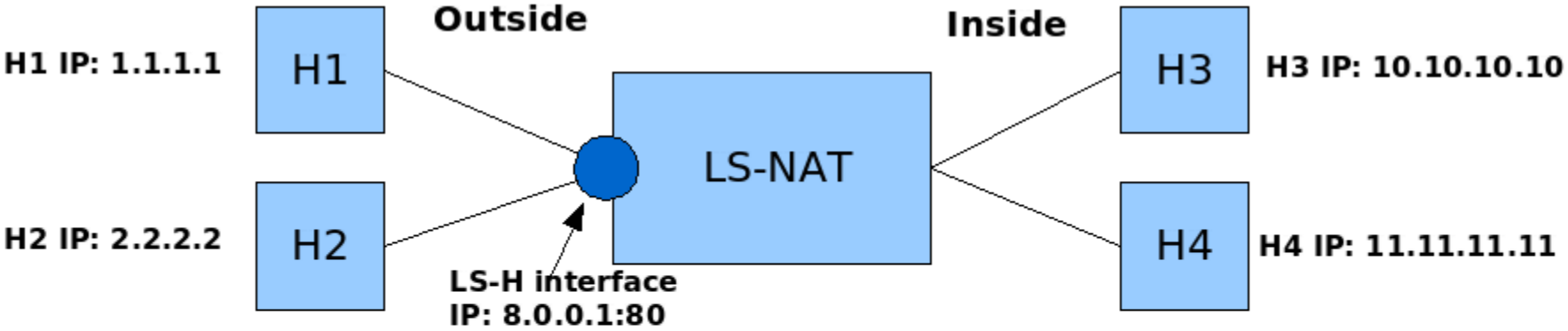
TCP H1 <- H4 (Outside interface)

Src-IP:	8.0.0.1
Dest-IP:	1.1.1.1
Src-Port:	40001
Dest-Port:	80

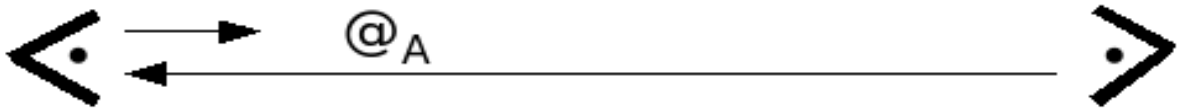
TCP H1 <- H4 (Inside interface)

Src-IP:	11.11.11.11
Dest-IP:	1.1.1.1
Src-Port:	40000
Dest-Port:	80

Load-sharing-NAT/NAPT 1/2



Entry	Outside-source	Outside-local	Inside-local
lsnat-H3	-	8.0.0.1:80	10.10.10.10:8000
lsnat-H4	-	8.0.0.1:80	11.11.11.11:8000
session-H1	1.1.1.1:40000	8.0.0.1:80	10.10.10.10:8000
session-H2	2.2.2.2:40000	8.0.0.1:80	11.11.11.11:8000



TCP H1 -> LS-H -> H3 (Outside interface)

Src-IP:	1.1.1.1
Dest-IP:	8.0.0.1
Src-Port:	40000
Dest-Port:	80

TCP H1 -> LS-H -> H3 (Inside interface)

Src-IP:	1.1.1.1
Dest-IP:	10.10.10.10.
Src-Port:	40000
Dest-Port:	8000

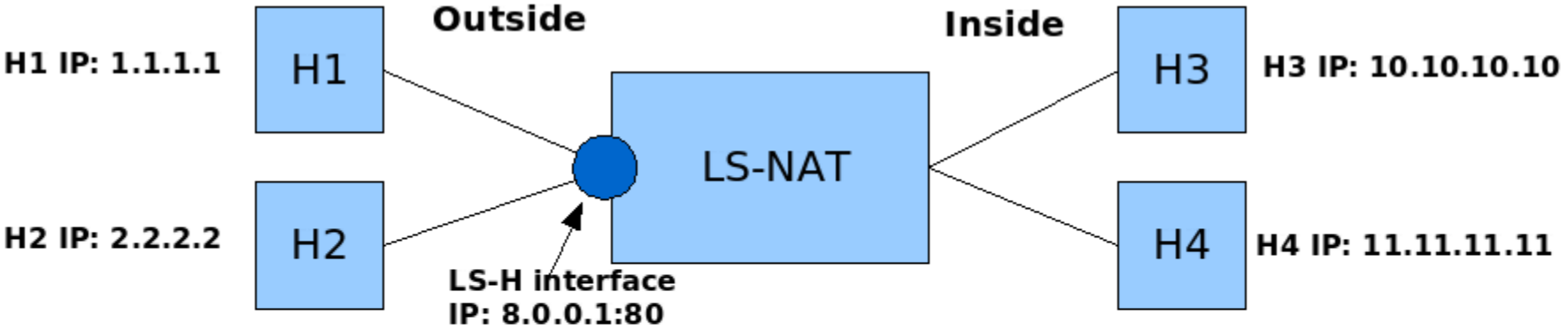
TCP H1 <- LS-H <- H3 (Outside interface)

Src-IP:	8.0.0.1
Dest-IP:	1.1.1.1
Src-Port:	80
Dest-Port:	40000

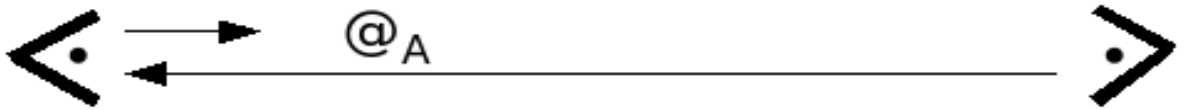
TCP H1 <- LS-H <- H3 (Inside interface)

Src-IP:	10.10.10.10
Dest-IP:	1.1.1.1
Src-Port:	8000
Dest-Port:	40000

Load-sharing-NAT/NAPT 2/2



Entry	Outside-source	Outside-local	Inside-local
lsnat-H3	-	8.0.0.1:80	10.10.10.10:8000
lsnat-H4	-	8.0.0.1:80	11.11.11.11:8000
session-H1	1.1.1.1:40000	8.0.0.1:80	10.10.10.10:8000
session-H2	2.2.2.2:40000	8.0.0.1:80	11.11.11.11:8000



TCP H2 -> LS-H -> H4 (Outside interface)

Src-IP:	2.2.2.2
Dest-IP:	8.0.0.1
Src-Port:	40000
Dest-Port:	80

TCP H2 -> LS-H -> H4 (Inside interface)

Src-IP:	2.2.2.2
Dest-IP:	11.11.11.11
Src-Port:	40000
Dest-Port:	8000

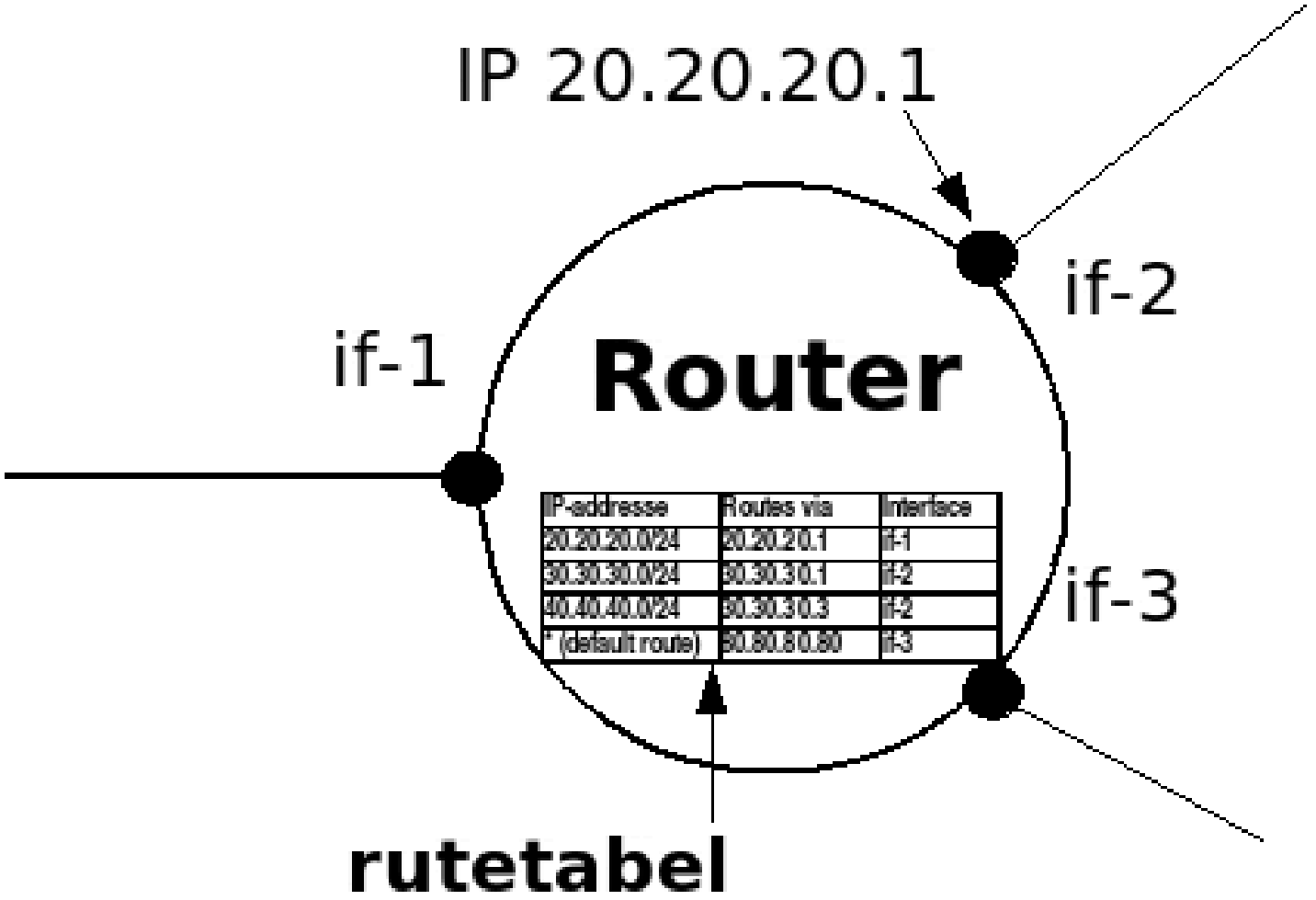
TCP H2 <- LS-H <- H4 (Outside interface)

Src-IP:	8.0.0.1
Dest-IP:	2.2.2.2
Src-Port:	80
Dest-Port:	40000

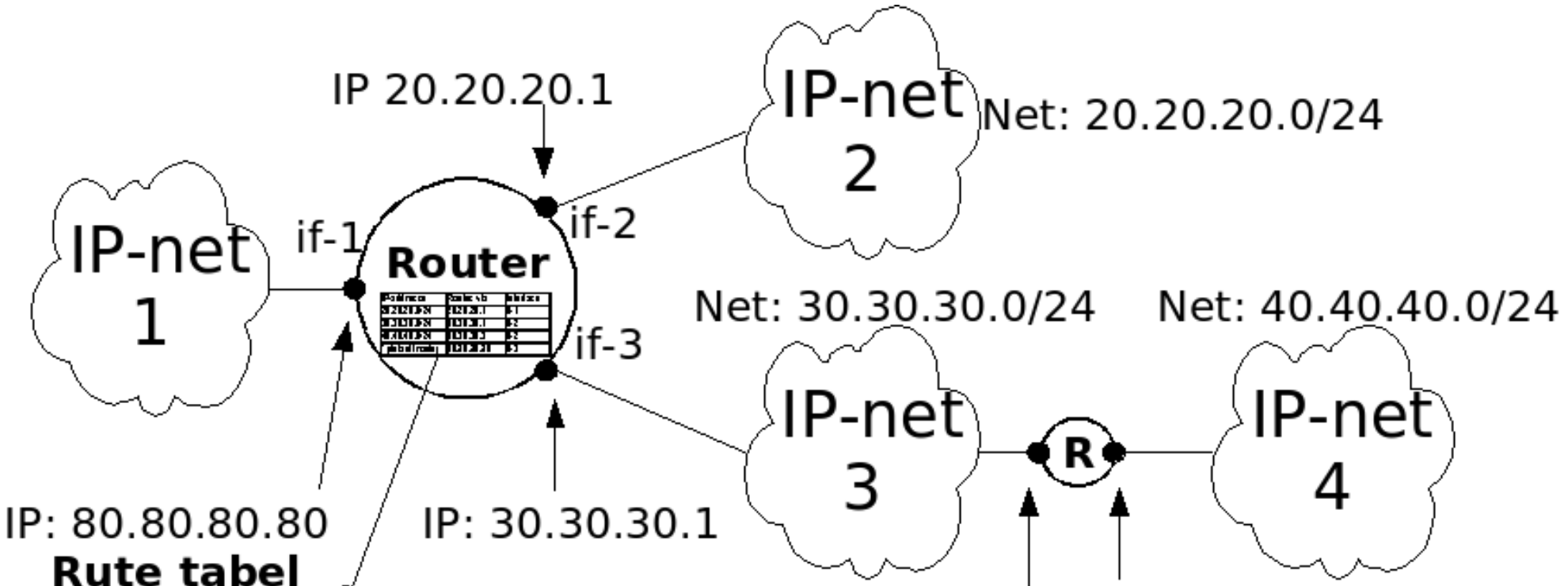
TCP H2 <- LS-H <- H4 (Inside interface)

Src-IP:	11.11.11.11
Dest-IP:	2.2.2.2
Src-Port:	8000
Dest-Port:	40000

Introduktion: IP-router



Introduktion: IP-router



Rute tabel

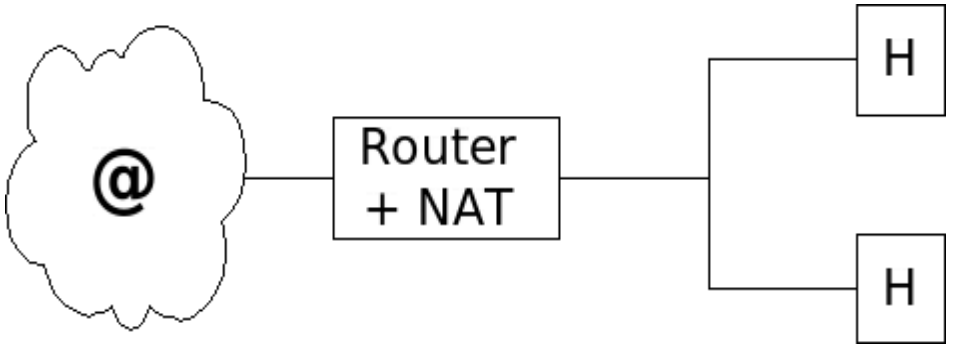
IP-adresse	Rutes via	Interface
20.20.20.0/24	20.20.20.1	if-2
30.30.30.0/24	30.30.30.1	if-3
40.40.40.0/24	30.30.30.2	if-3
* (default rute)	80.80.80.80	if-1

Analyse af andre NAT-implementationer. 1/3

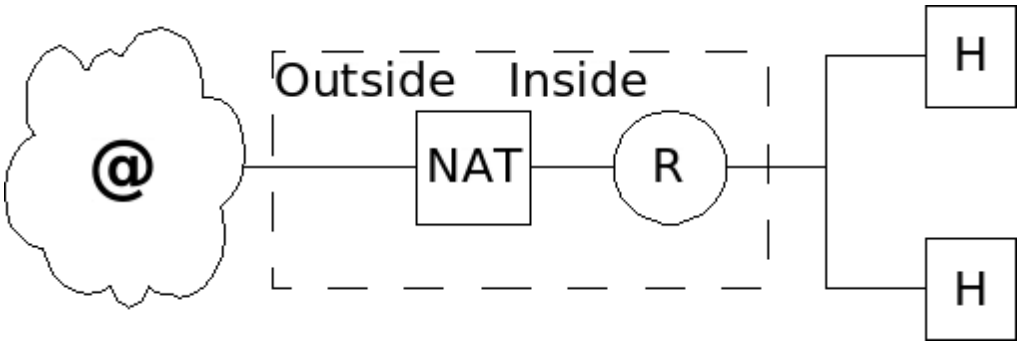
- Cisco, Juniper, FreeBSD natd.
- Analyse af:
  - Hvor er NAT-funktionen placeret i routeren?
  - Er det muligt (og en fordel) med mere end 2 NAT-interfaces (sider) per NAT-enhed?
  - Få inspiration fra konfigurationssprogene
  - Åbenlyse fordele/begrænsninger ved hver enkelt implementation

Analyse af andre NAT-implementationer. 2/3
Hvor sidder ”den” præcis.

En router med indbygget NAT



Cisco og FreeBSD definerer begge at NAT-delen sidder imellem det ”udvendige” interface og rutningsfunktionen:(Juniper nævner ikke noget om dette)



Analyse af andre NAT-implementationer. 3/3

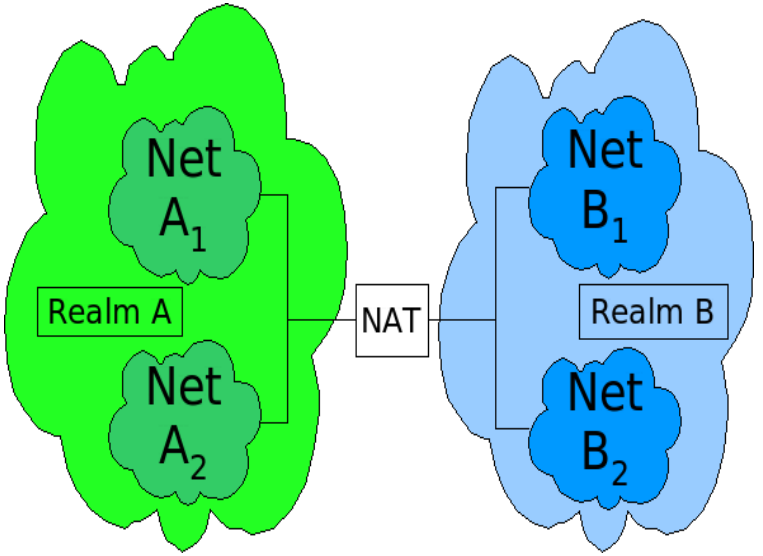
- Ingen af de valgte versioner har mulighed for mere end 2 sidet NAT-interfaces.
- Sprogforskelle
  - Cisco: Tagger interfaces med ”inside” og ”outside”, god support for time-out indstillinger i sproget (ved time-out fjernes NAT-tabel regler)
  - Juniper: Anvender kun IP definitionen til dette.
  - FreeBSD: Har LS-NAT som den eneste af de 3. Er meget begrænset mht. konfigurerbarhed ved dynamisk nat.

XORP-NAT-konfigurationssprog 6/9
Forslag til forbedringer

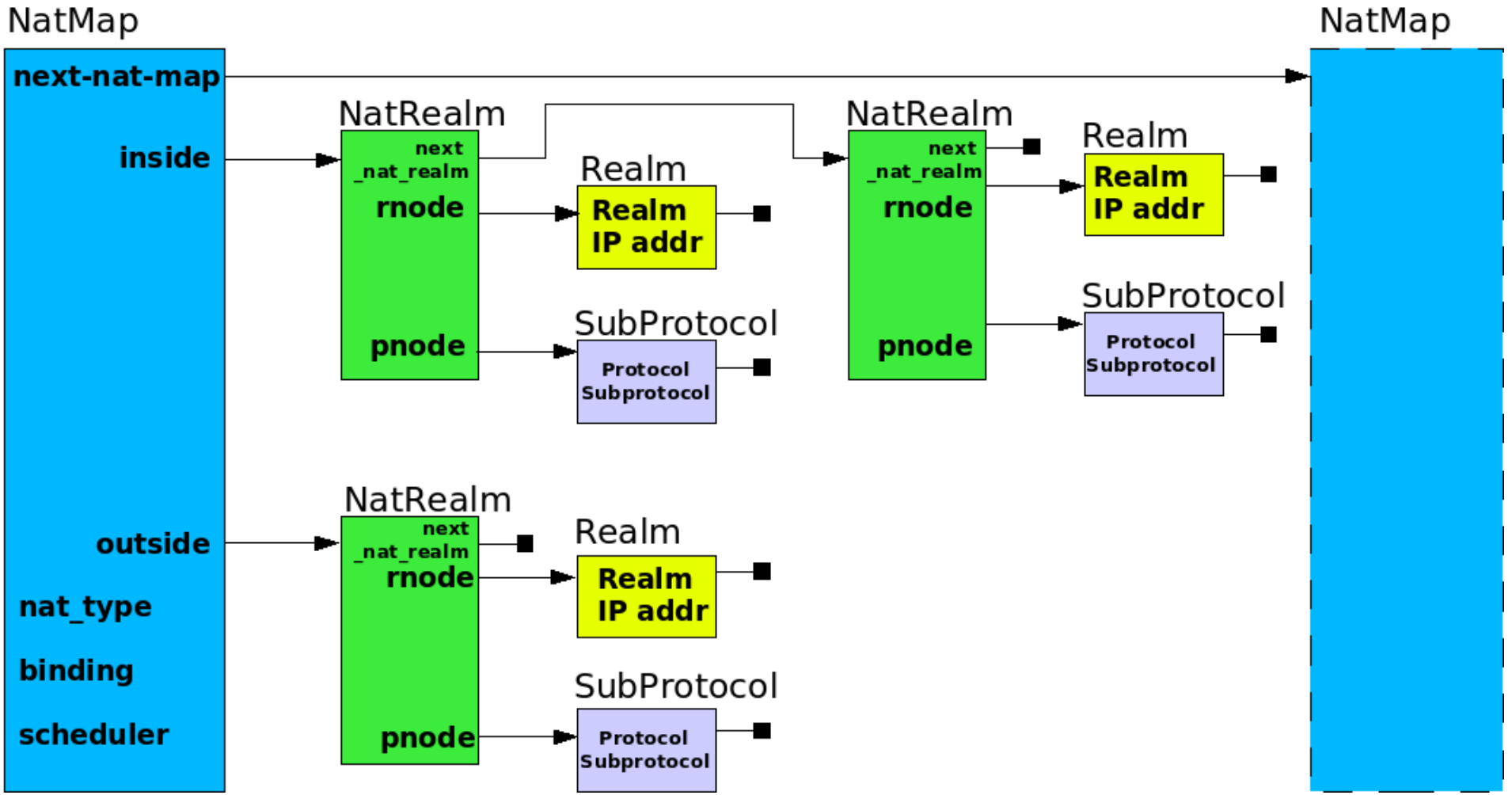
- Nødvendigt med mange parametre per NAT-interface.
- Muligt at reducere antallet af parametre per NAT-interface ved at anvende en defineret syntax repræsenteret i en tekststreng.
- Fortolkning i NAT-modulet er nødvendig for at udtrække konfigurations data og type informationen.

XORP-NAT-konfigurationssprog 1/8

- NAT-moduler har 2 sider vi kalder dem hhv ”inside” og ”outside”.
- Mange mulige parametre på hver side af NAT-modulet.



Implementering af XORP-NAT-konfigurationsdatalager 4/5



The END