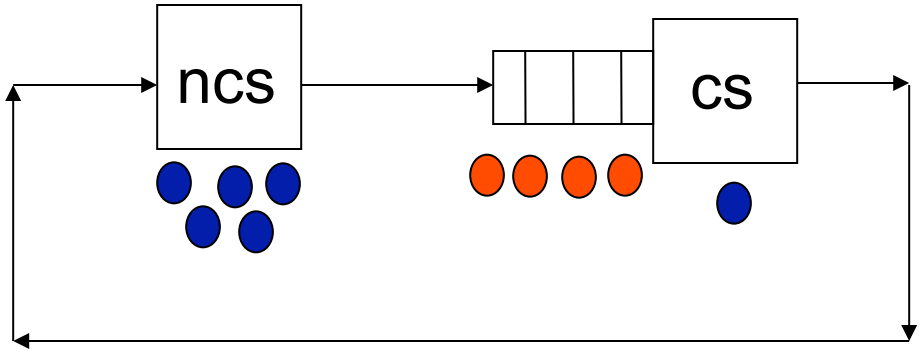
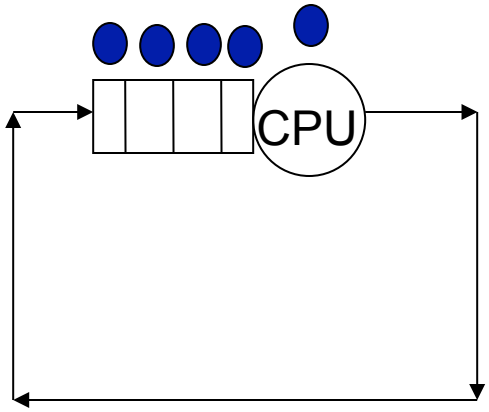
Two-Level Iterative Queuing Modeling of Software Contention

Daniel A. Menascé
Dept. of Computer Science
George Mason University
www.cs.gmu.edu/faculty/menasce.html

Motivation

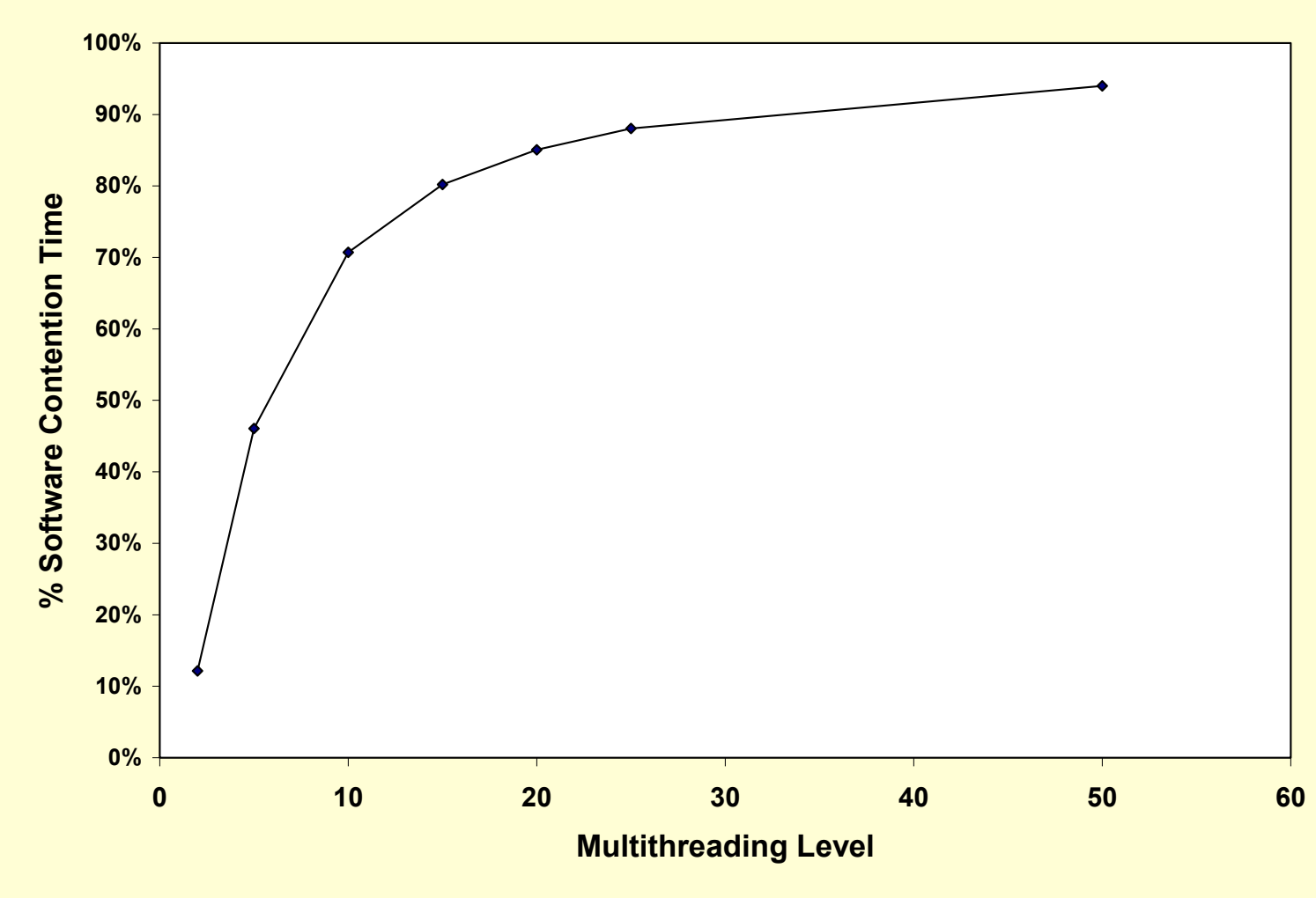


Software View

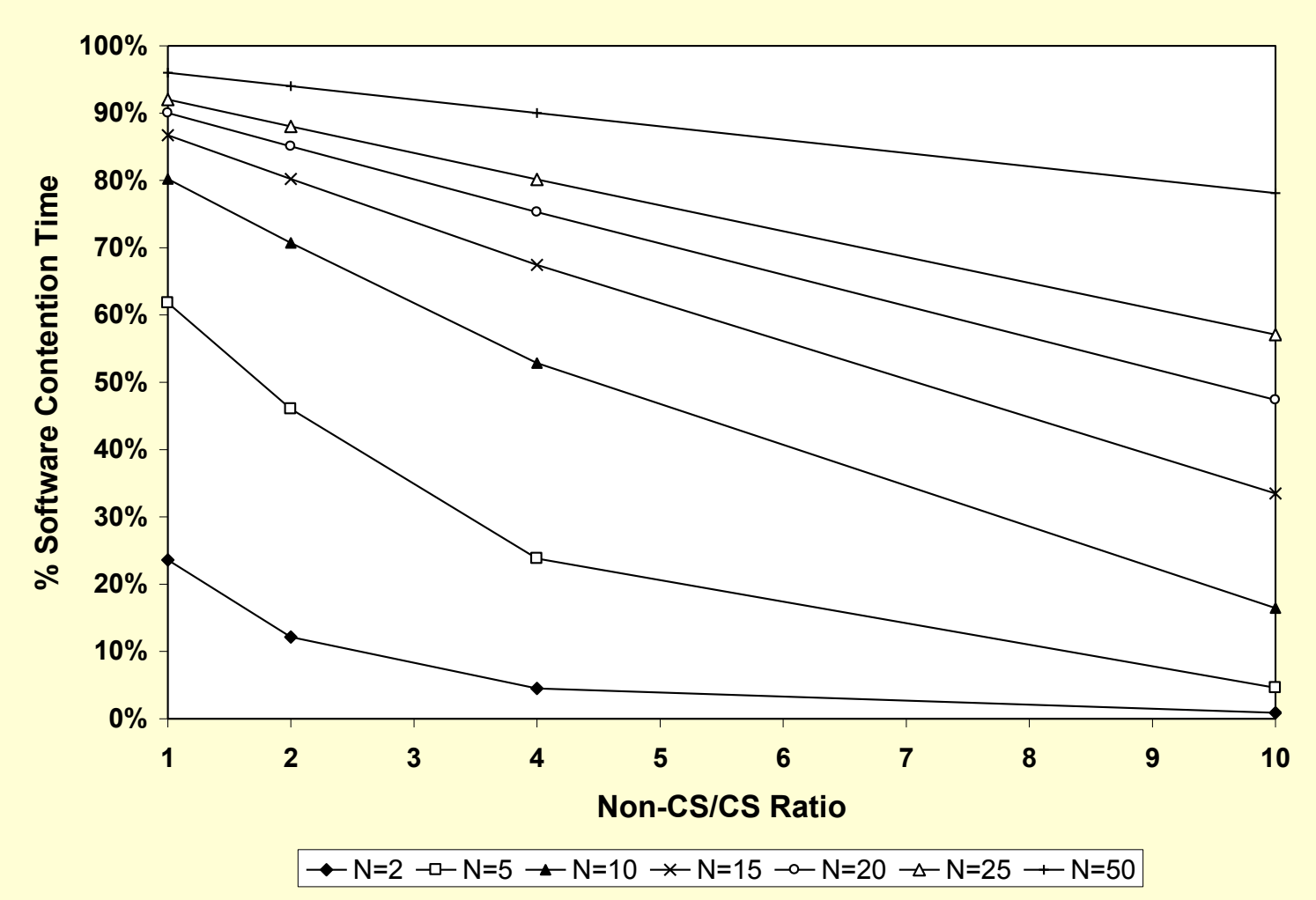


Hardware View

% Software Contention Time vs. Multithreading Level

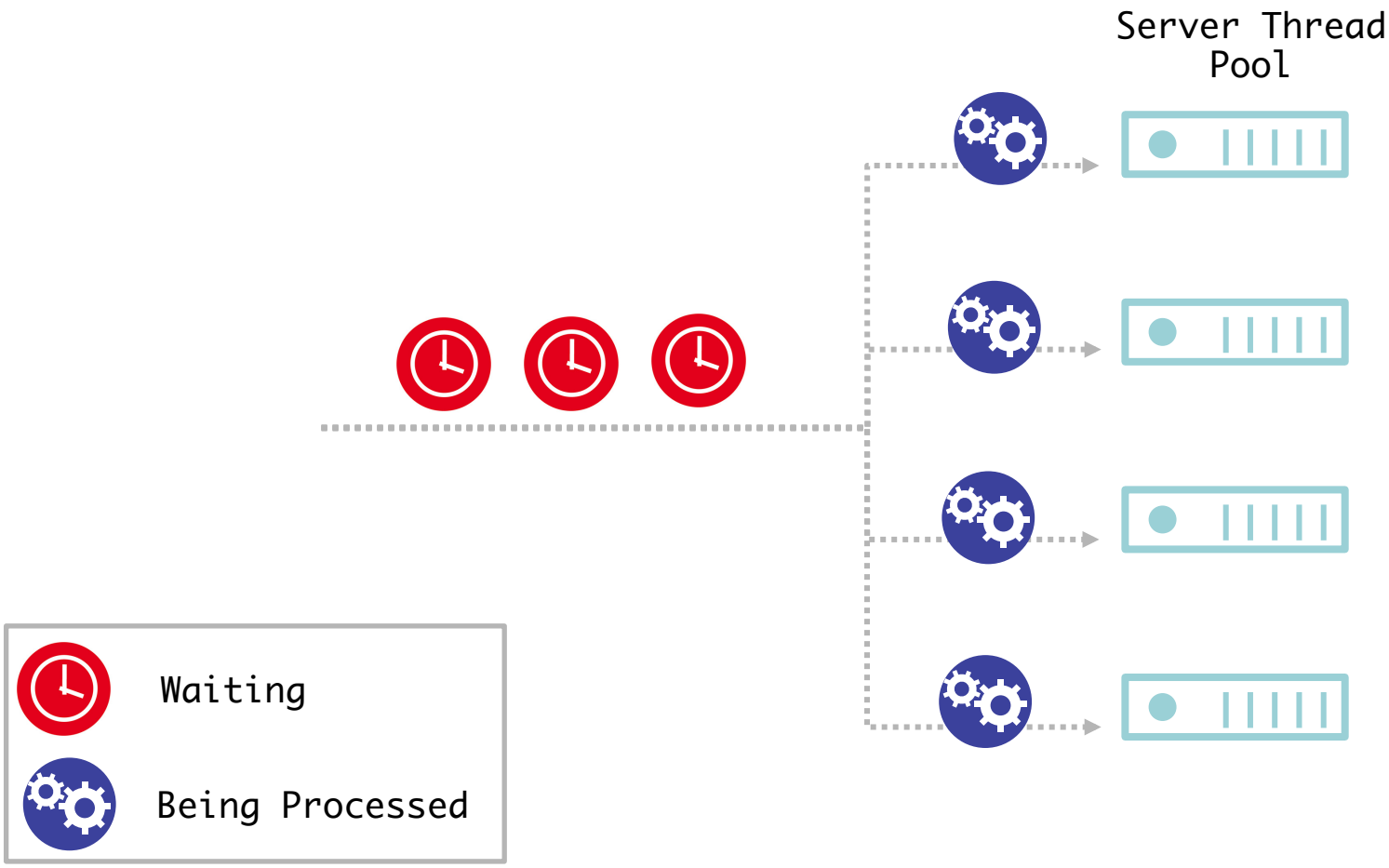


%Sofware Contention Time vs Non-CS/CS Ratio



Software Contention

example: multithreaded software server



QN Models that Capture SW Contention

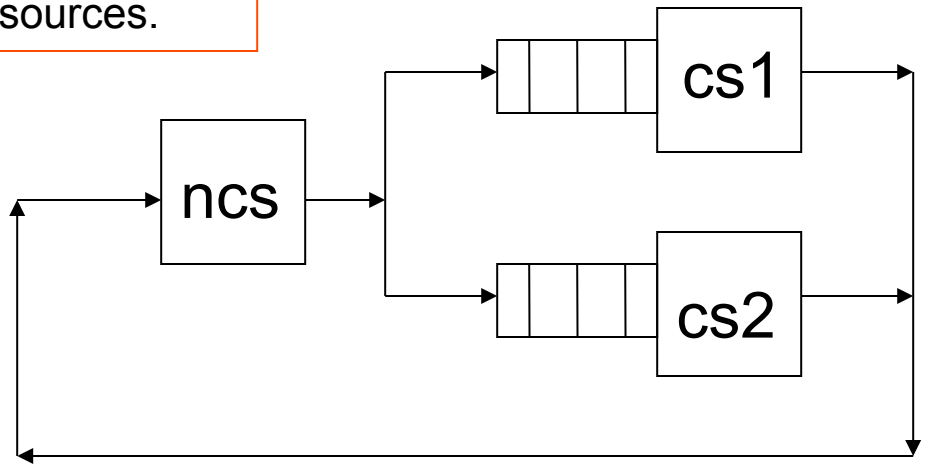
1. Carleton University's Layered Queuing Networks (LQN)
   - Neilson, J.E., C.M. Woodside, D.C. Petriu, and S. Majumdar, Software bottlenecking in client-server systems and rendezvous networks, IEEE Tr. Software Engineering, 21(9), 1995.
   - Rolia, J. and K.C. Sevcik, The Method of Layers, IEEE Tr. Software Engineering, 21(8), 1995, pp.689–700.
   - Franks, G., T. Al-Omari, C.M. Woodside, O. Das, and S. Derisavi, Enhanced Modeling and Solution of Layered Queueing Networks, IEEE Tr. Software Engineering, 35(2), 2009.
2. Menascé's two-level iterative SQN-HQN model
   - Menascé, D.A., Two-level Iterative Queuing Modeling of Software Contention, Proc. Tenth IEEE/ACM Intl. Symp. Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2002),Fort Worth, TX, Oct. 12-16, 2002.
   - Menascé, D.A., V.A.F. Almeida, and L.W. Dowdy. Performance by Design: Computer Capacity Planning By Example, Prentice Hall, 2004.

QN Models that Capture SW Contention

1. Carleton University's Layered Queuing Networks (LQN)
   - Neilson, J.E., C.M. Woodside, D.C. Petriu, and S. Majumdar, Software bottlenecking in client-server systems and rendezvous networks, IEEE Tr. Software Engineering, 21(9), 1995.
   - Rolia, J. and K.C. Sevcik, The Method of Layers, IEEE Tr. Software Engineering, 21(8), 1995, pp.689–700.
   - Franks, G., T. Al-Omari, C.M. Woodside, O. Das, and S. Derisavi, Enhanced Modeling and Solution of Layered Queueing Networks, IEEE Tr. Software Engineering, 35(2), 2009.
2. Menascé's two-level iterative SQN-HQN model
   - Menascé, D.A., Two-level Iterative Queuing Modeling of Software Contention, Proc. Tenth IEEE/ACM Intl. Symp. Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2002),Fort Worth, TX, Oct. 12-16, 2002.
   - Menascé, D.A., V.A.F. Almeida, and L.W. Dowdy. Performance by Design: Computer Capacity Planning By Example, Prentice Hall, 2004.

If we just consider the SQN we are ignoring time spent at the software resources due to contention for hardware resources.

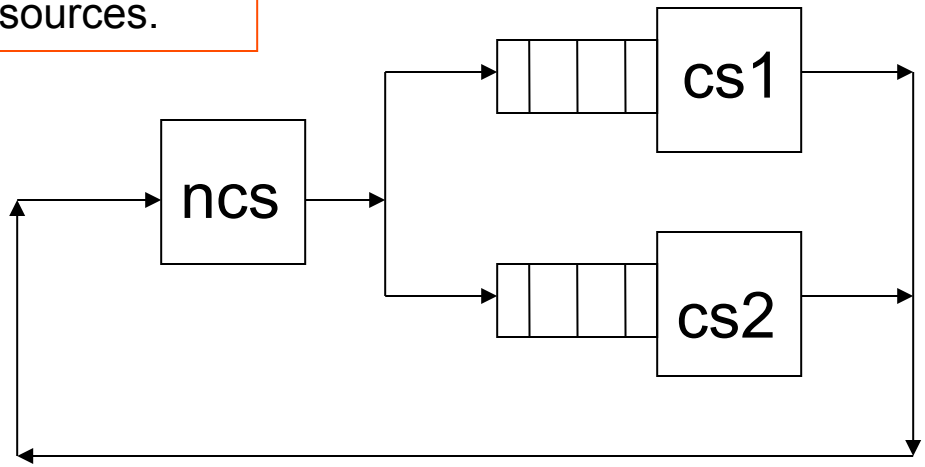
Approach



Software QN

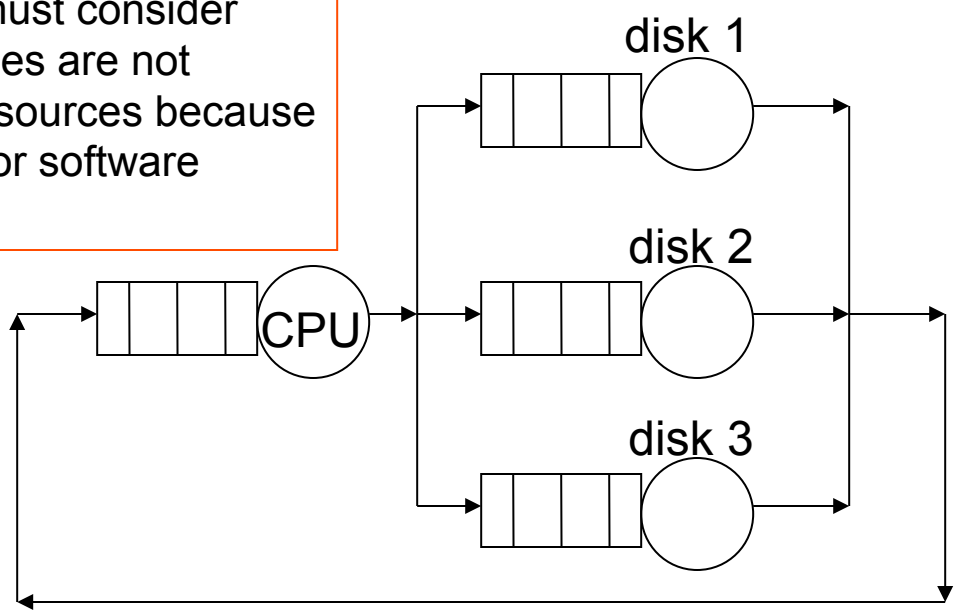
Approach

If we just consider the SQN we are ignoring time spent at the software resources due to contention for hardware resources.



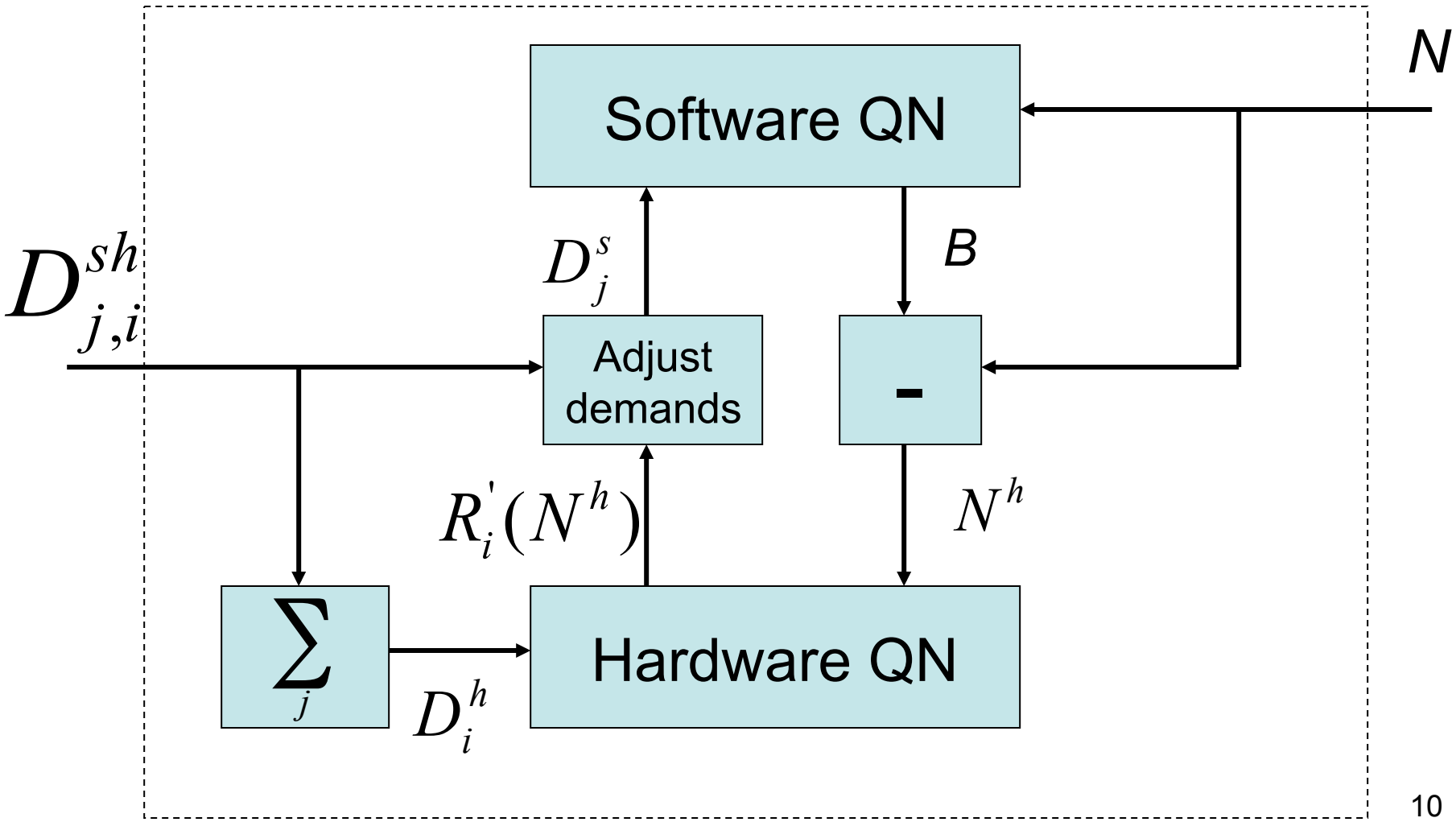
Software QN

The HQN model must consider that some processes are not using hardware resources because they are blocked for software resources.



Hardware QN

SQN-HQN Scheme



Input Service Demands

	Software Modules			
Hardware Devices (↓)	**NCS**	**CS 1**	**CS 2**	**Hardware Demands**
CPU	0.2000	0.0600	0.0808	0.3408
Disk 1	0.0560	0.0576	0.0000	0.1136
Disk 2	0.0360	0.0000	0.1212	0.1572
Disk 3	0.0360	0.0000	0.0000	0.0360
Software Demands	0.3280	0.1176	0.2020	

sum across a column

sum across a row

Service Demands

	Software Modules			
Hardware Devices (↓)	**NCS**	**CS 1**	**CS 2**	**Hardware Demands**
CPU	0.2000	0.0600	0.0808	0.3408
Disk 1	0.0560	0.0576	0.0000	0.1136
Disk 2	0.0360	0.0000	0.1212	0.1572
Disk 3	0.0360	0.0000	0.0000	0.0360
Software Demands	0.3280	0.1176	0.2020	

$$D^{s}_{cs1}$$

(total service time of a software module)

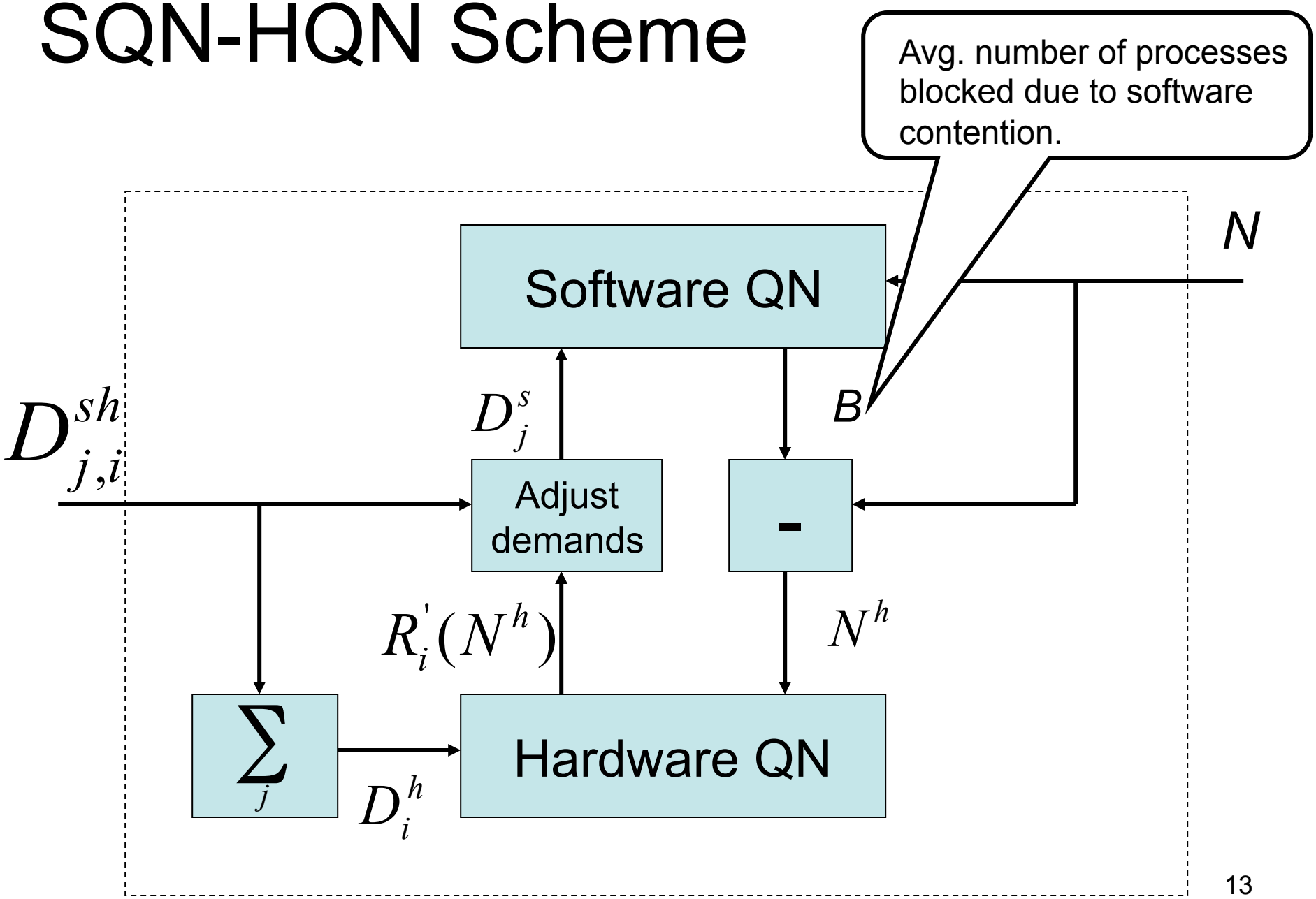
$$D^{sh}_{cs1,disk1}$$

(total service time of a software module at a physical device)

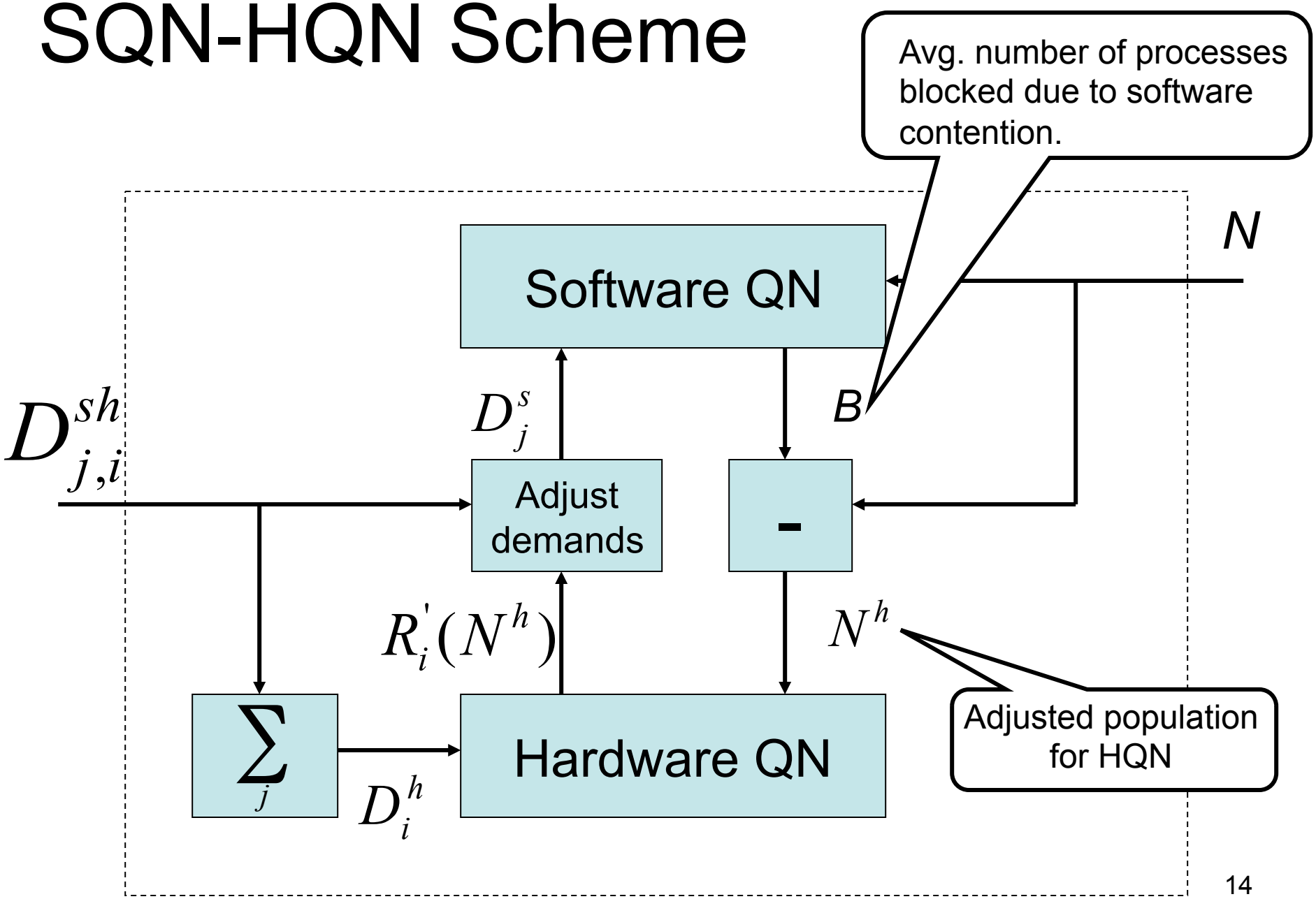
$$D^{h}_{disk1}$$

(total service time of the application at a given device)

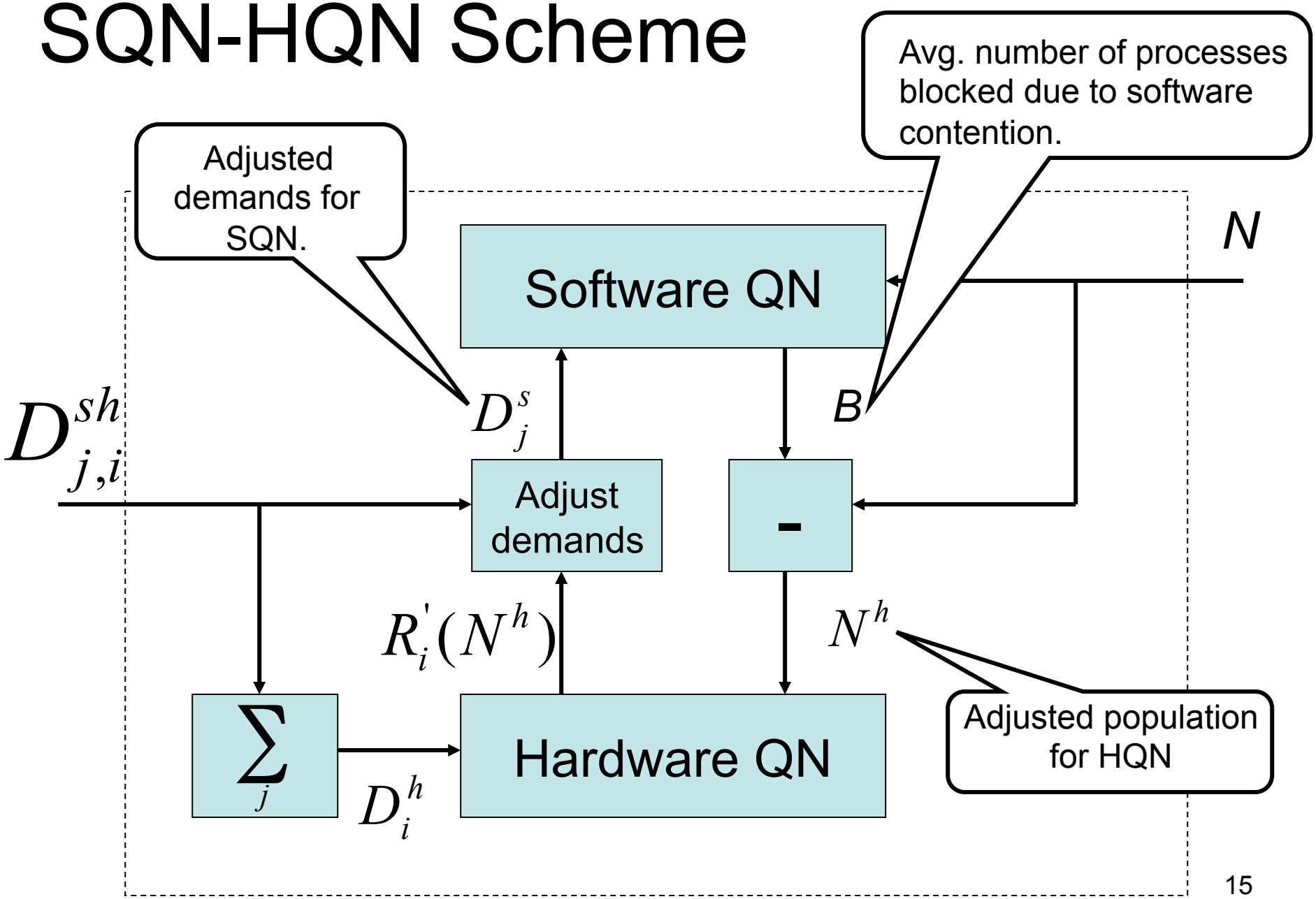
SQN-HQN Scheme



SQN-HQN Scheme



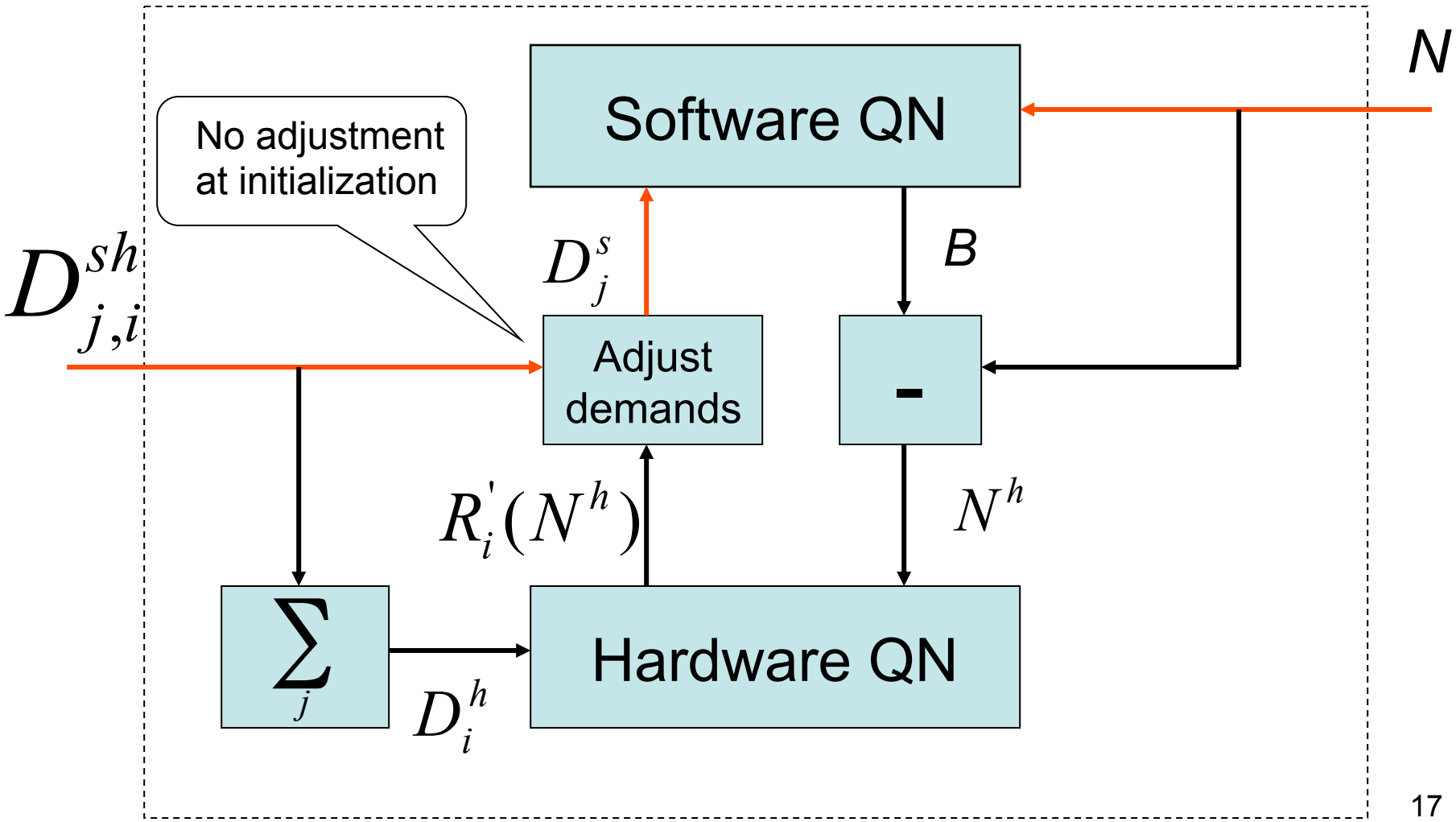
SQN-HQN Scheme



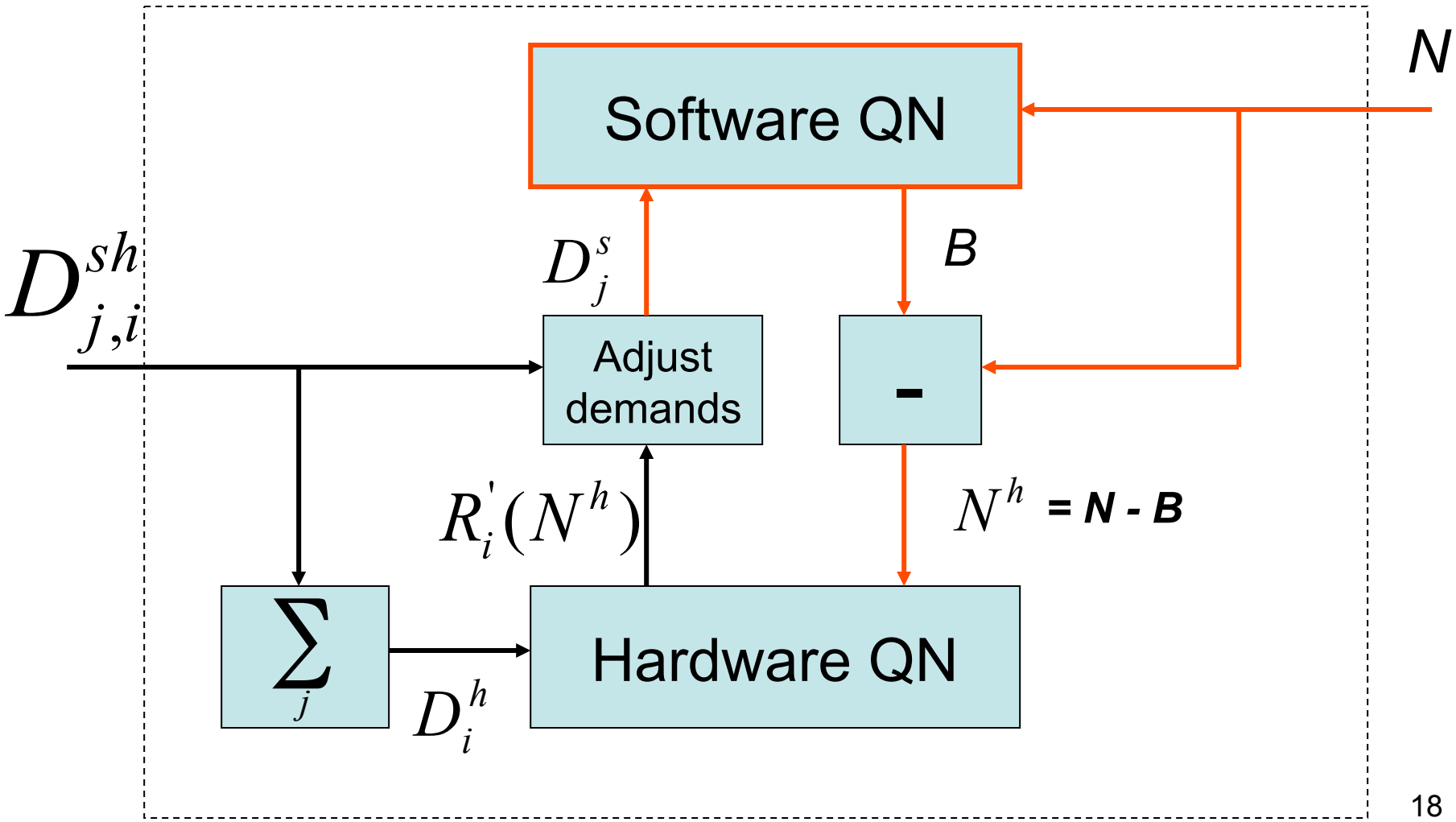
Basic Idea

- Iteration between solving the SQN and the HQN.
- Number B of processes blocked due to software contention computed through the SQN.
- Population at HQN is reduced by B.
- Service demands at SQN are adjusted to account for physical contention.

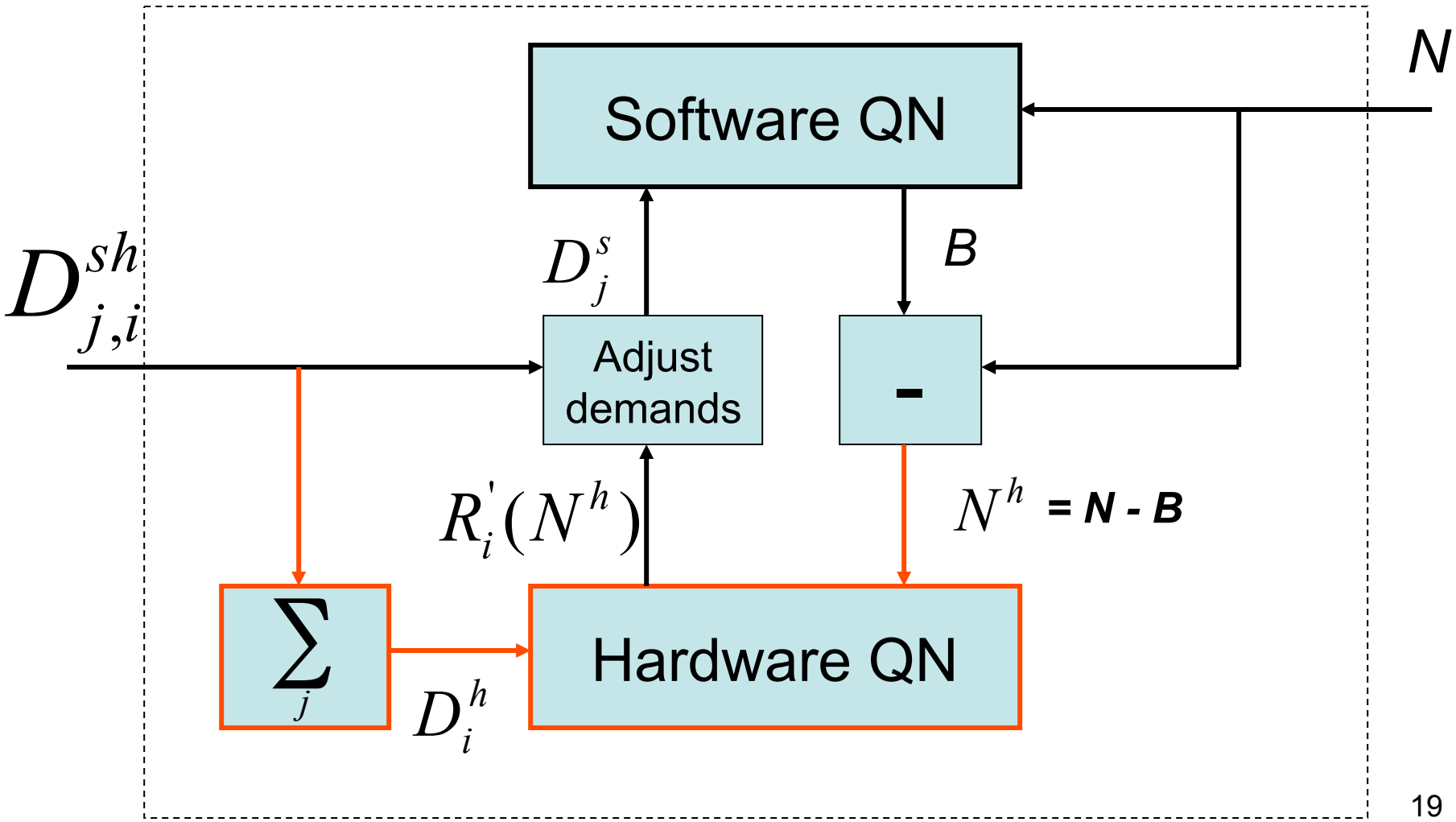
SQN-HQN: Initialization



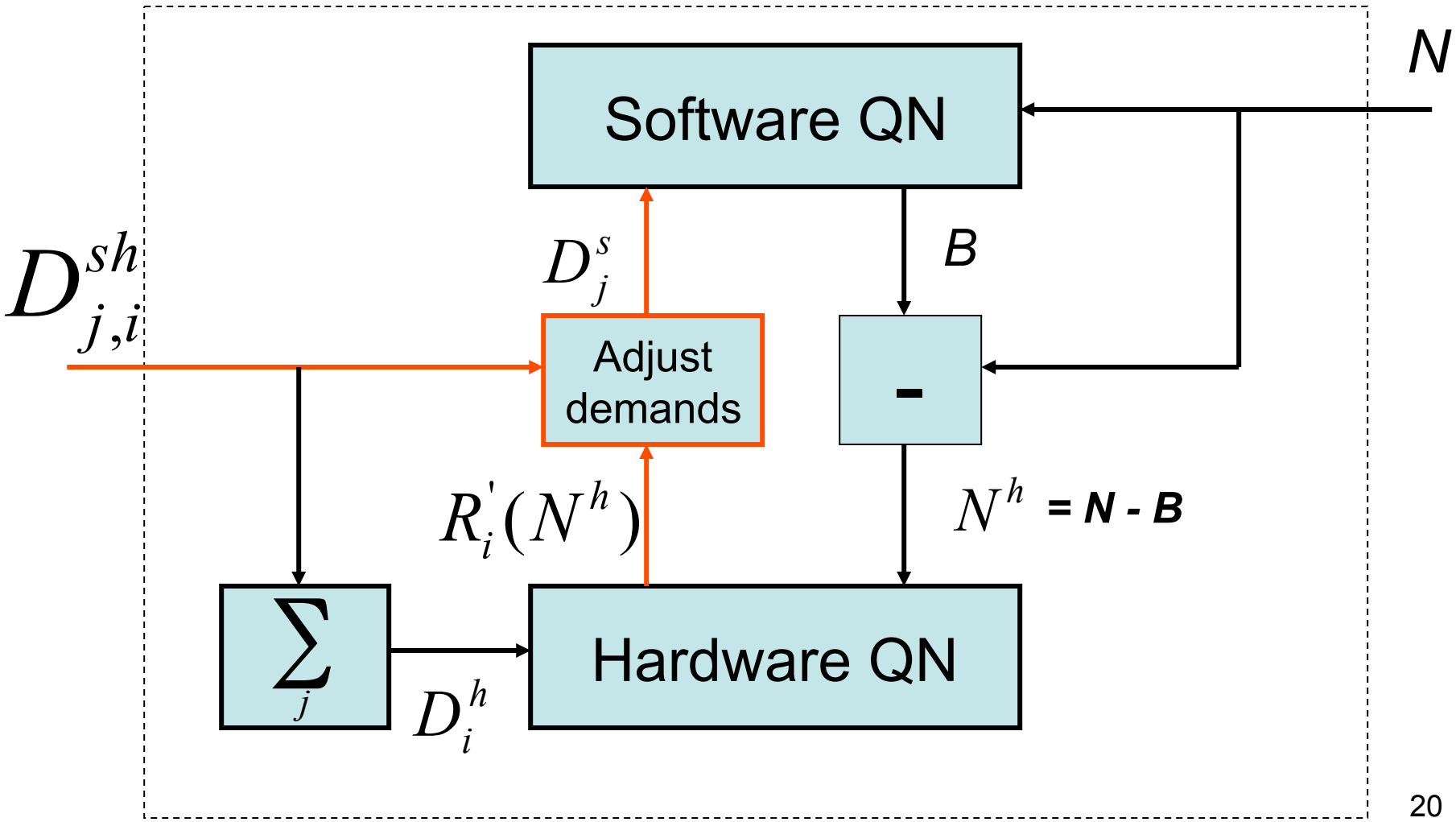
SQN-HQN: Solve SQN – No Hardware Contention



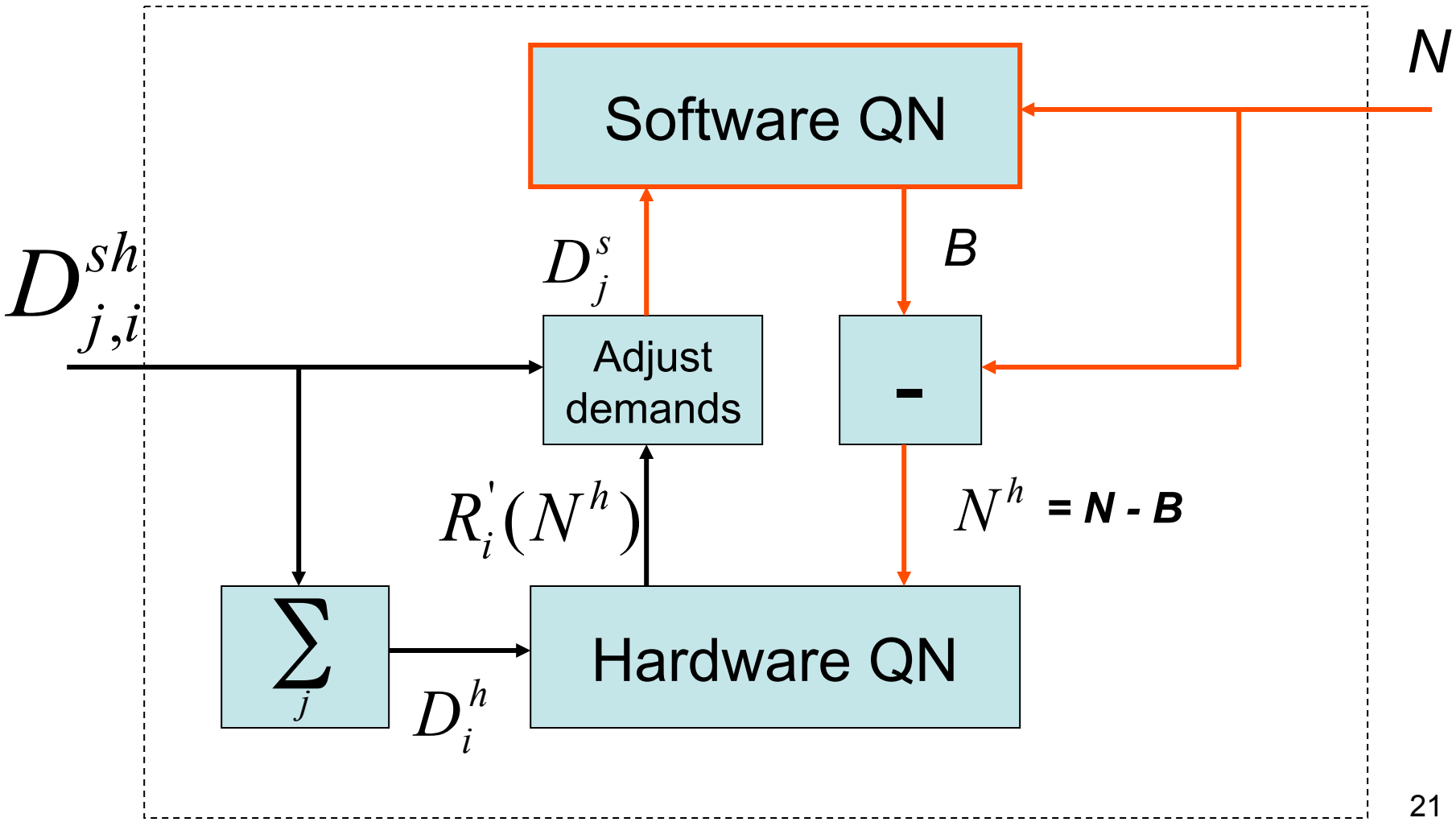
SQN-HQN: Solve HQN



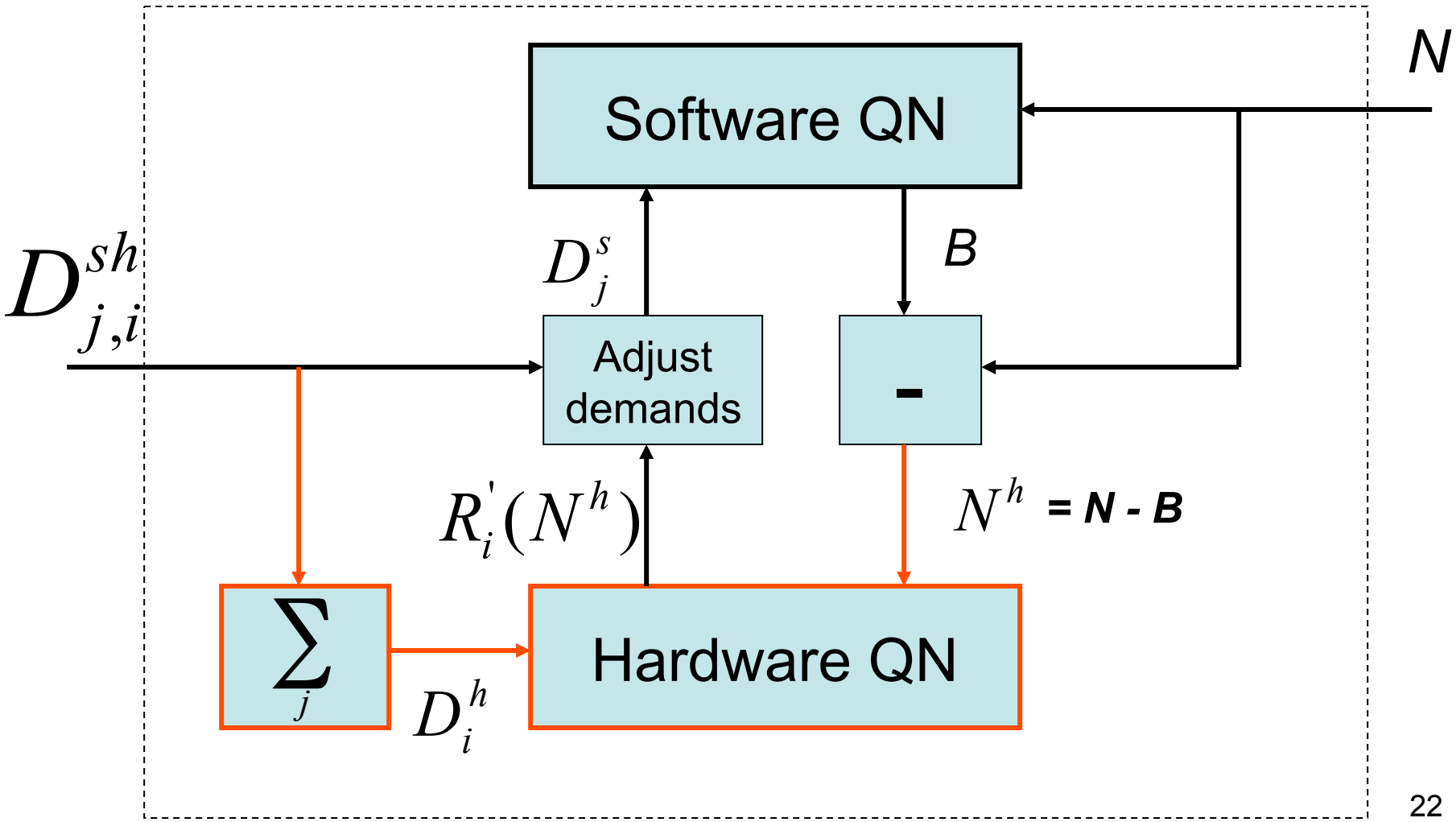
SQN-HQN: Adjust demands for SQN



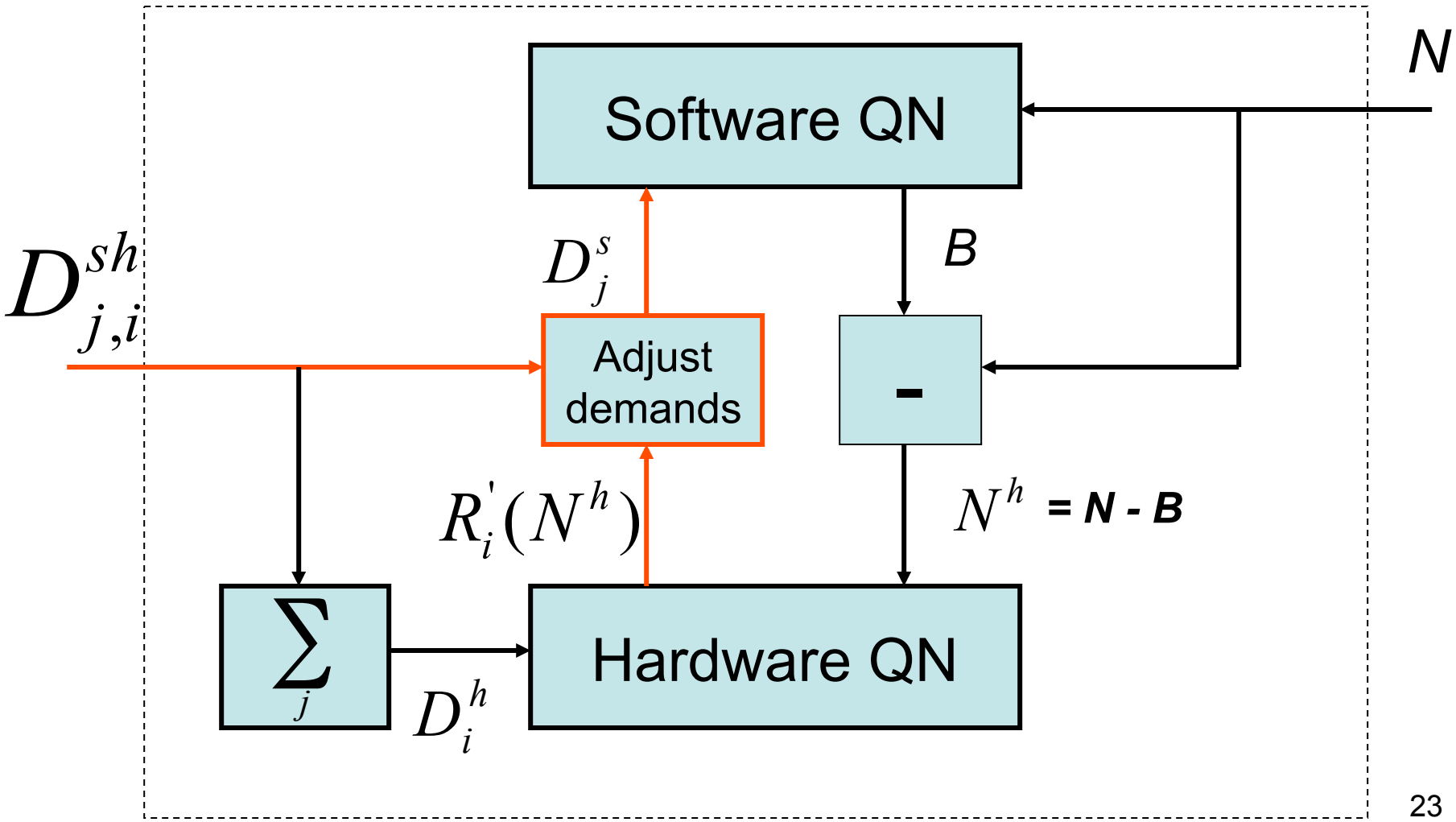
SQN-HQN: Solve SQN Again



SQN-HQN: Solve HQN



SQN-HQN: Adjust demands for SQN



and so on …

Convergence is checked on absolute relative error on the number of blocked processes in the SQN.

Adjustment of SQN Demands

- Single class case:

residence time at device i.

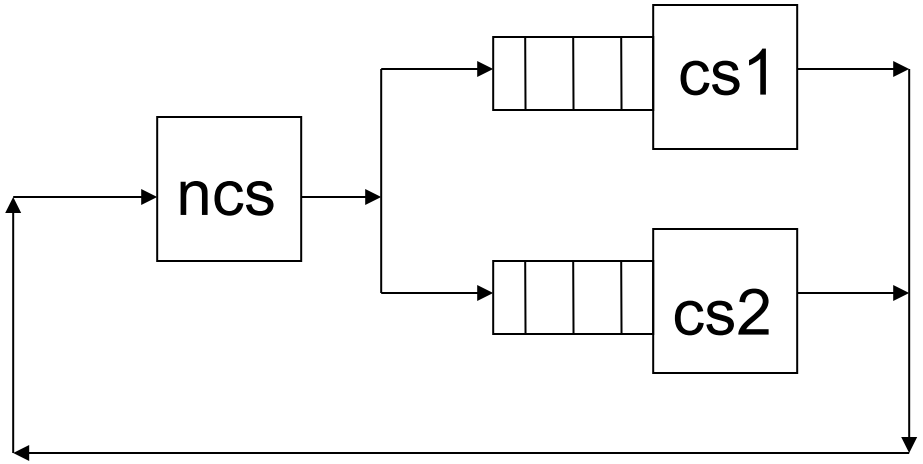
$$D_j^s \leftarrow \sum_i \frac{D_{j,i}^{sh}}{D_i^h} \times R_i^{'}(N^h)$$

- Multiple class case:

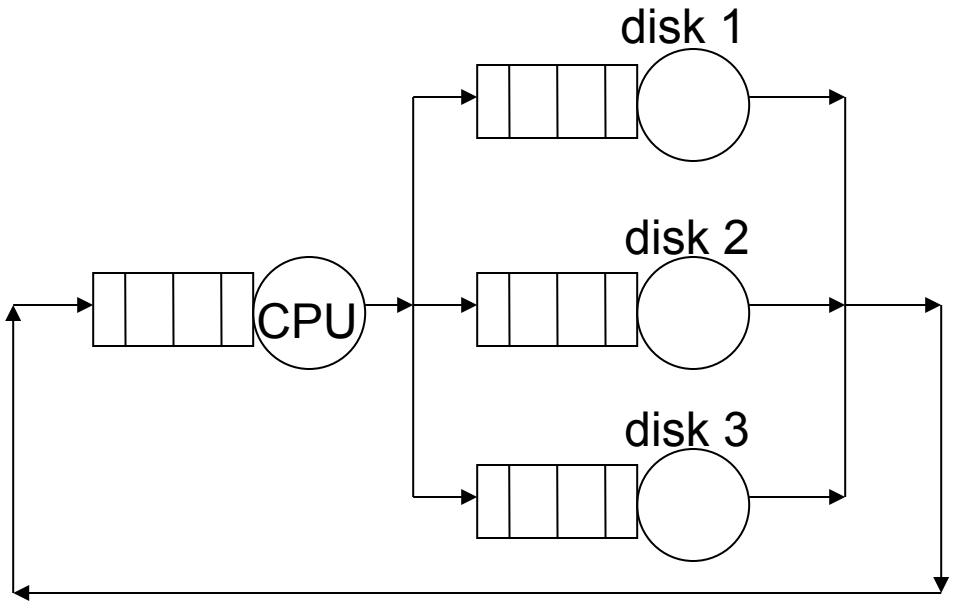
residence time at device i. for class r.

$$D_{j;r}^s \leftarrow \sum_i \frac{D_{j;i,r}^{sh}}{D_{i;r}^h} \times R_{i;r}^{'}(\vec{N}^h)$$

Example



Software QN



Hardware QN

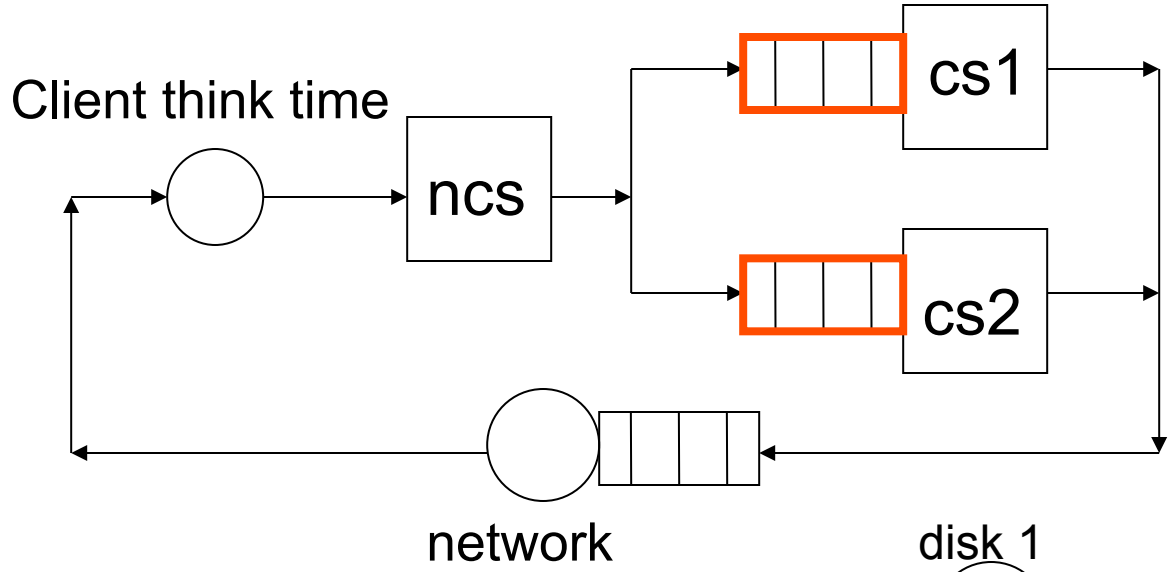
Comparison with other approaches

			Absolute % Error		
N	**SQN-HQN**	**GB**	**SQN_HQN**	**ASM**	**ASPA**
1	1.544	1.54	0.27	0.00	
2	2.088	2.11	1.06	4.60	
3	2.317	2.37	2.22	5.89	4.2
4	2.428	2.49	2.49	5.99	2.8
5	2.487	2.56	2.86	5.87	2.0
6	2.521	2.60	3.05	5.78	1.5
7	2.541	2.62	3.00	5.75	1.5
8	2.555	2.63	2.86	5.77	1.1

GB: global balance equations
ASM: Aggregate Server Method [Agrawal and Buzen 1983]
ASPA: Avg. Subsystem Population Approximation [Jacobson & Lazowska 1983]

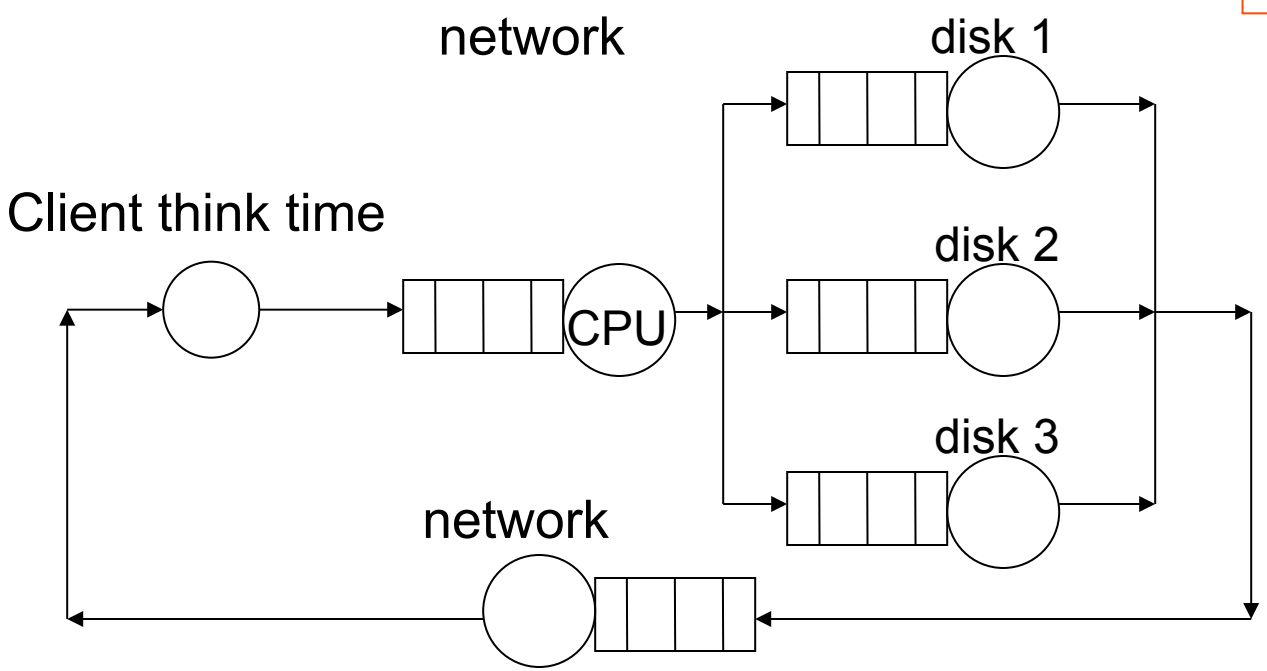
SQN is consistently pessimistic.
ASPA is much more complex to implement.
ASM only works for one class.

Modeling Non-Software Resources



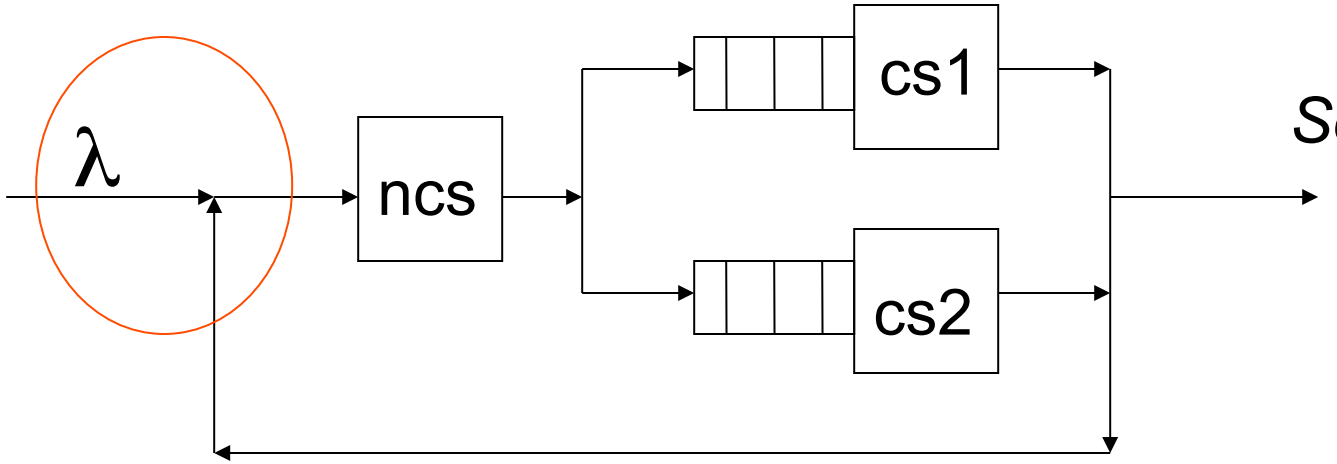
Software QN

B is the avg. no. of processes in the software resource waiting lines.

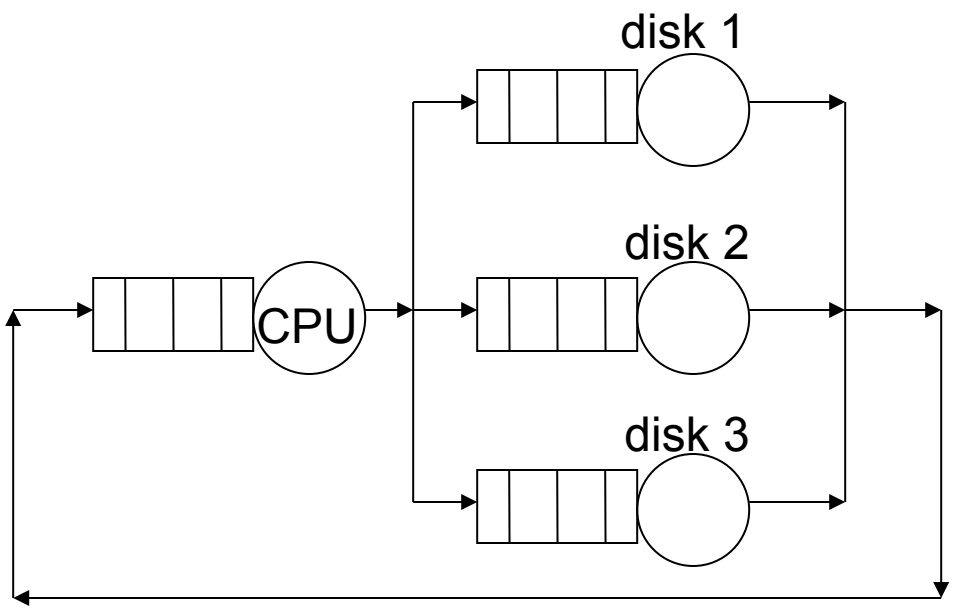


Hardware QN

Open QN at the Software Level

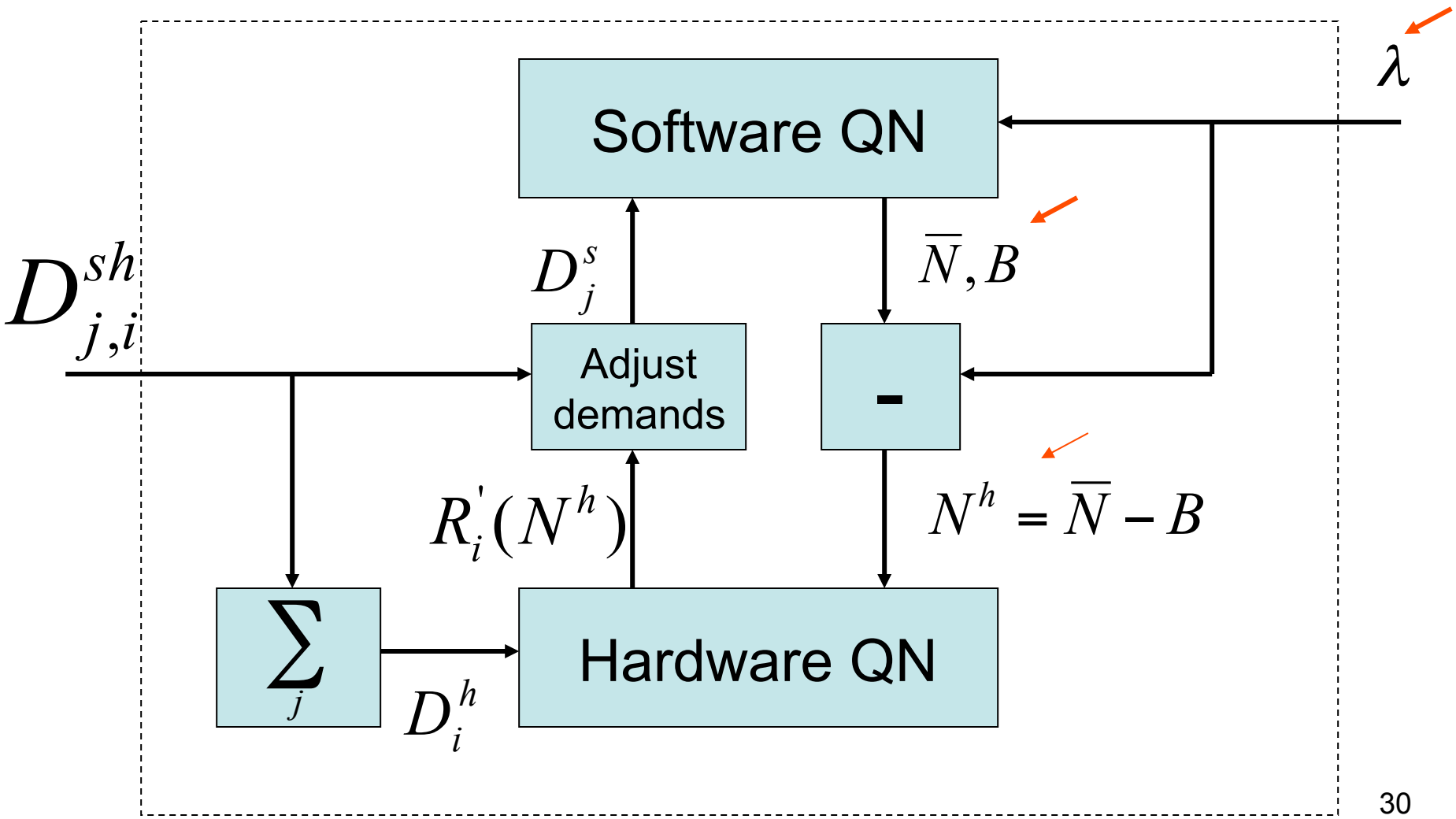


Software QN



Hardware QN

SQN-HQN Scheme: Open SQN



Results of Iterations for Open SQN Case

					Adjusted SQN Demands		
Iteration	**Nh**	**Resp. Time**	**Ns**	**B**	**NCS**	**CS1**	**CS2**
0	-				0.3280	0.1176	0.2020
1	1.295	0.821	1.641	0.346	0.3662	0.1302	0.2235
2	1.440	0.946	1.893	0.453	0.3858	0.1365	0.2342
3	1.513	1.014	2.028	0.515	0.3958	0.1397	0.2396
4	1.550	1.050	2.100	0.549	0.4010	0.1414	0.2424
5	1.570	1.069	2.137	0.568	0.4037	0.1423	0.2438
6	1.580	1.079	2.157	0.577	0.4051	0.1427	0.2446
7	1.585	1.084	2.167	0.582	0.4059	0.1430	0.2450
8	1.588	1.086	2.173	0.585	0.4062	0.1431	0.2452
9	1.589	1.088	2.175	0.587	0.4064	0.1432	0.2453
10	1.590	1.088	2.177	0.587			

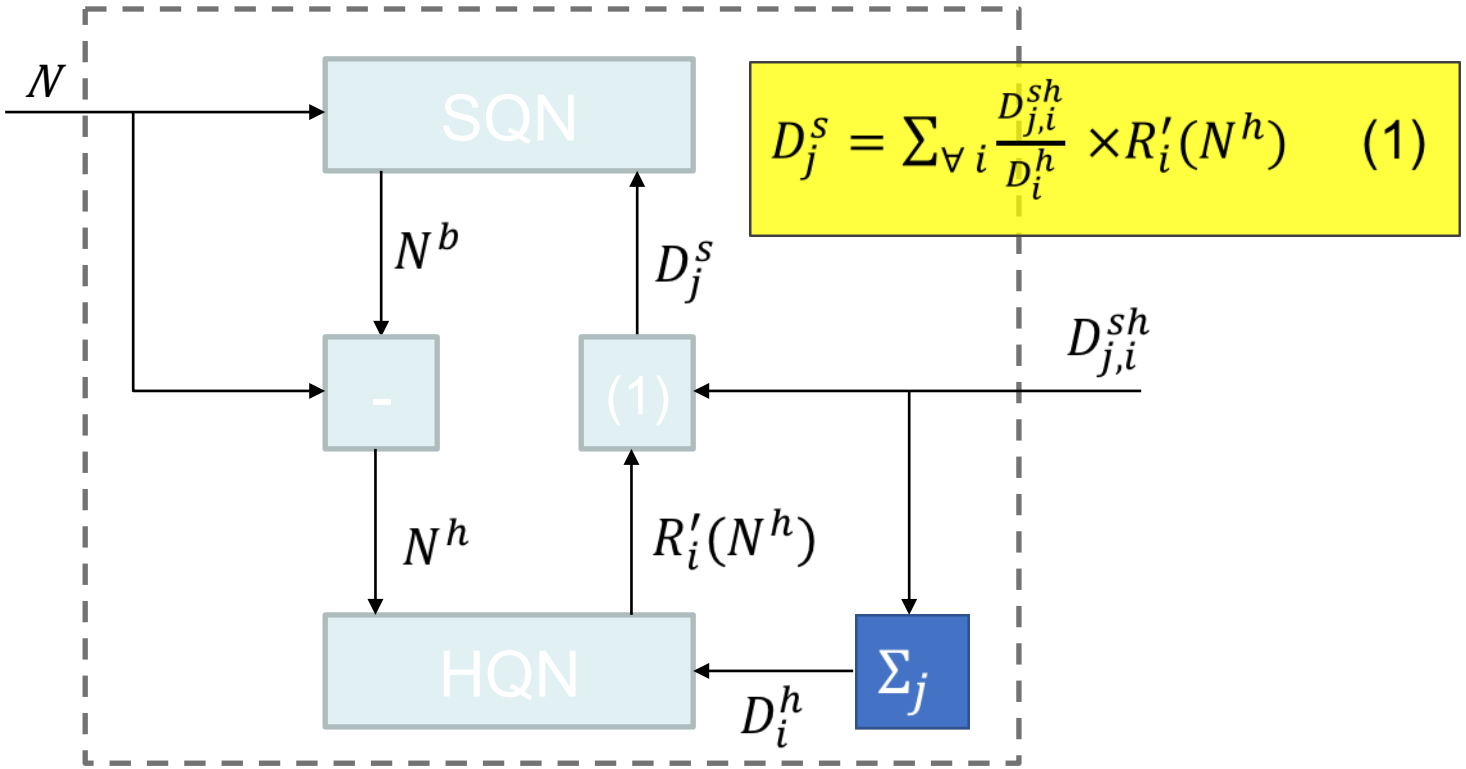
Modeling and Optimization of Multitiered Server-based Systems

The SQN-HQN Model

- SQN
  - the software queueing network
  - models the software application using a software network
- HQN
  - the hardware queueing network
  - models the hardware infrastructure on top of which the software runs

- Software modules blocked at the SQN are not counted as active in the HQN.
- Contention computed by the HQN is used to elongate the execution time of software modules at the SQN.

The SQN-HQN Model

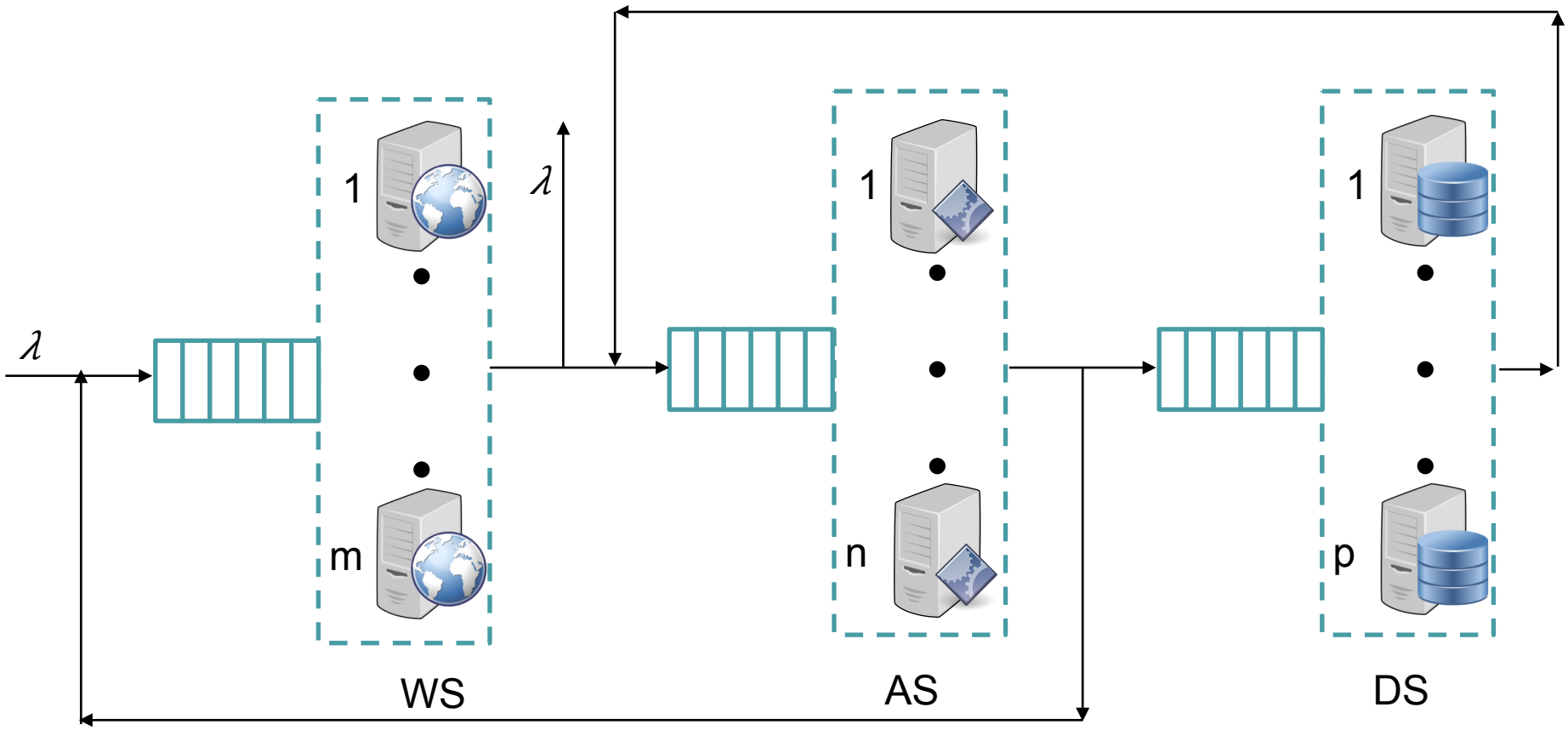


$$D_j^s = \sum_{\forall i} \frac{D_{j,i}^{sh}}{D_i^h} \times R_i'(N^h) \qquad (1)$$

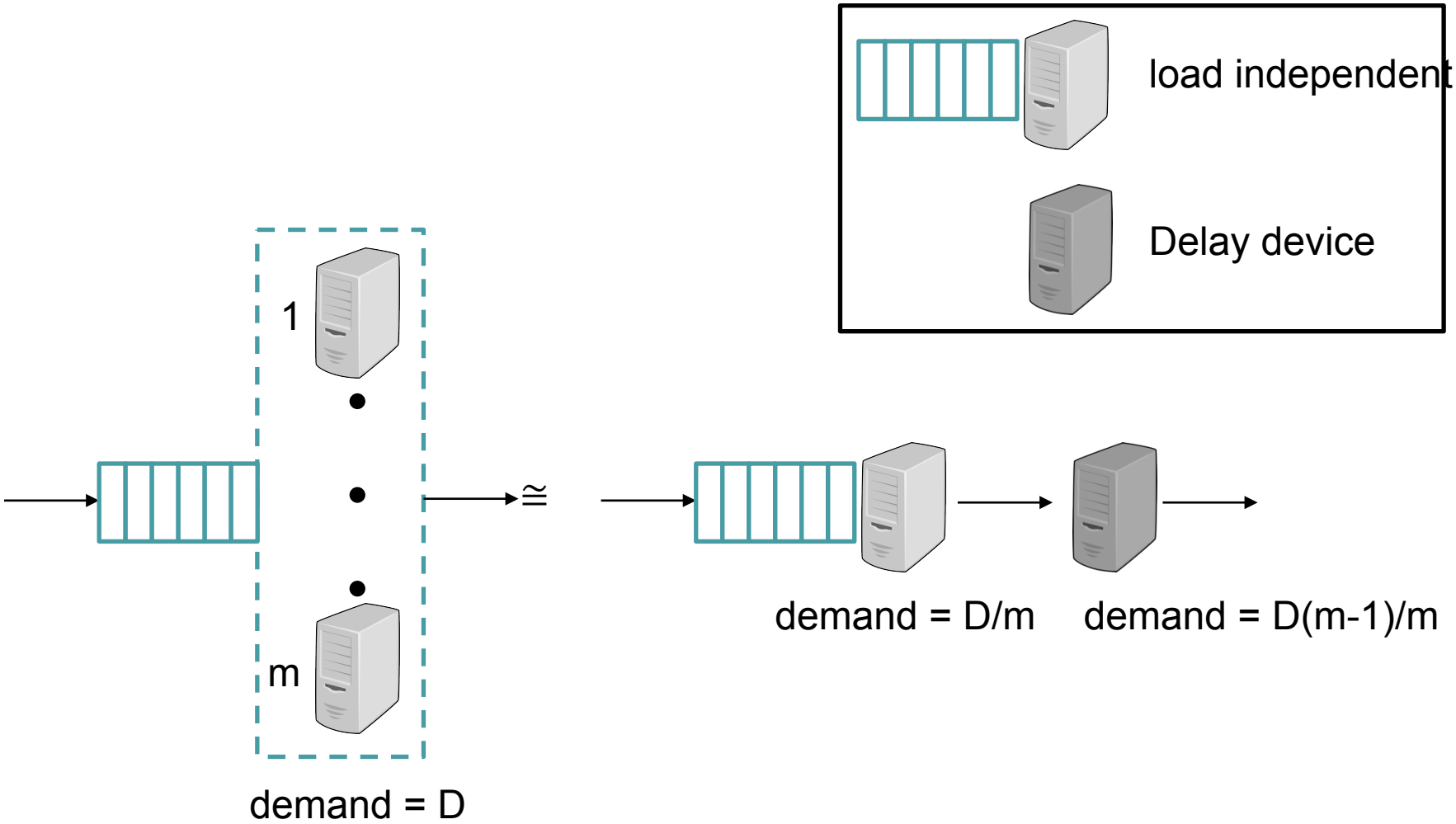
$$while \left| N_k^b - N_{k-1}^b \right| < \epsilon$$

A 3-tiered Server-based System: the **SQN**

single-queue multiple-servers / Open



Seidmann's Approximation to Model Multi-server Queues



Residence time at SQN

$$R = \begin{cases} \dfrac{D}{1-\lambda D} & load\ independent \\ \\ D & delay \end{cases}$$

$$R_{SQN} = \frac{D_{ws}/m}{1-\lambda\, D_{ws}/m} + \frac{D_{ws}(m-1)}{m} +$$

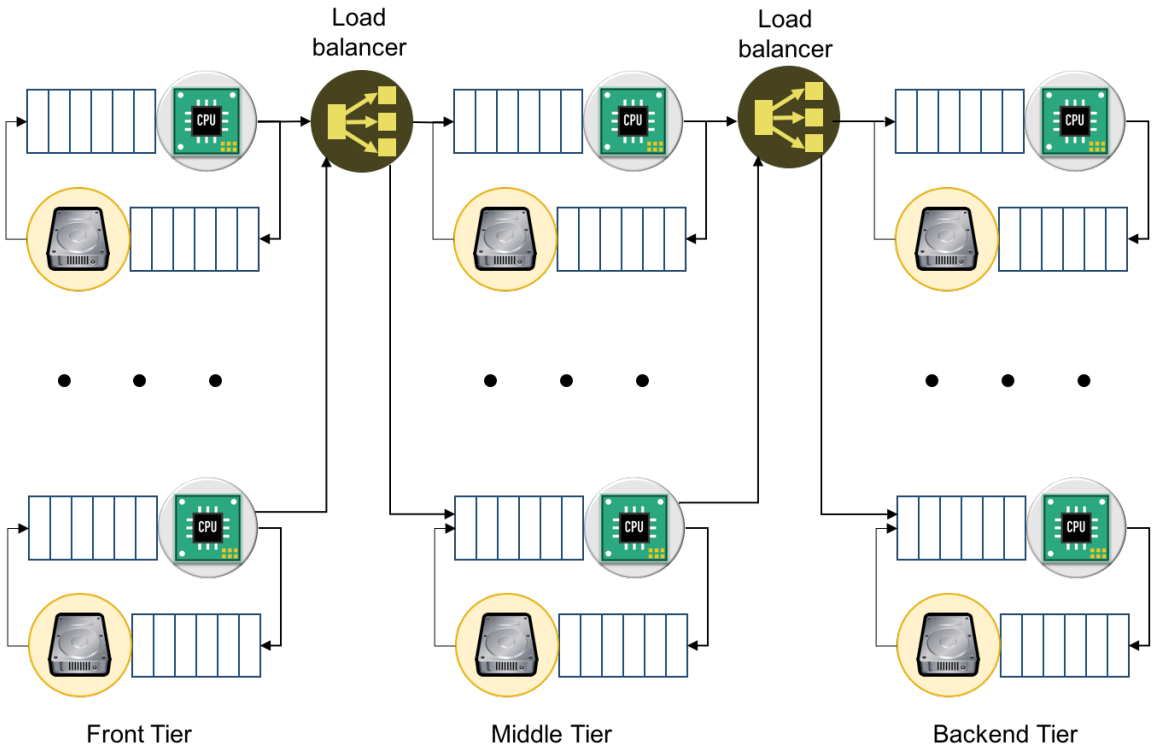
$$\frac{D_{as}/n}{1-\lambda\, D_{as}/n} + \frac{D_{as}(n-1)}{n} +$$

$$\frac{D_{ds}/p}{1-\lambda\, D_{ds}/p} + \frac{D_{ds}(p-1)}{p}$$

A 3-tiered Server-based System: the **HQN**

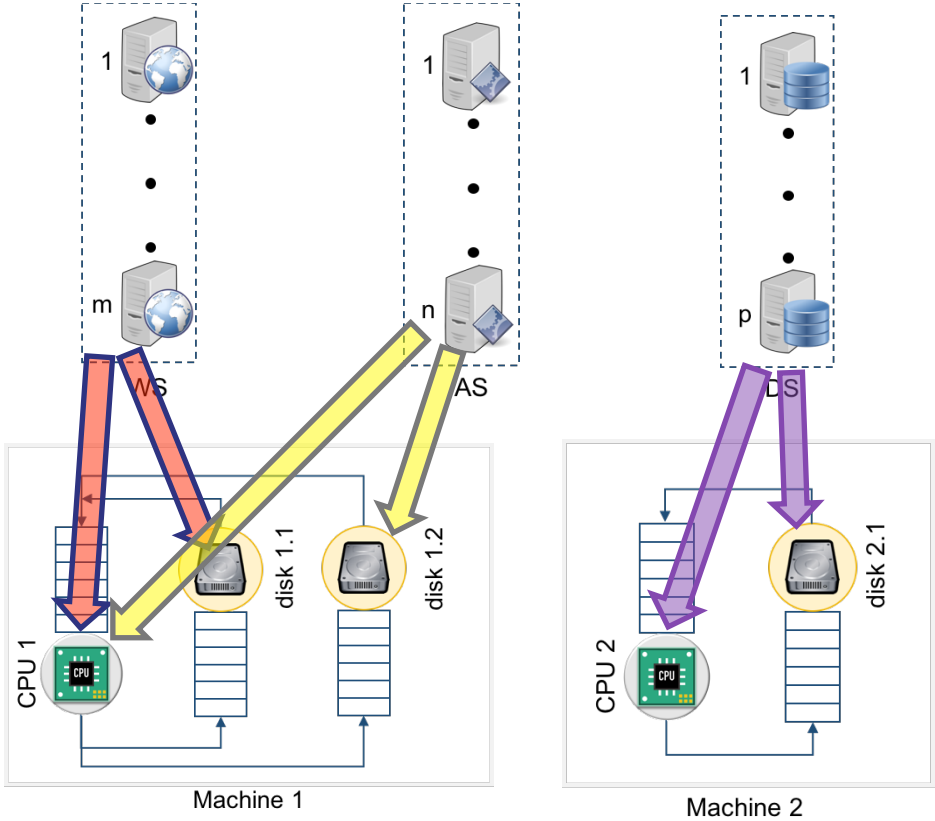
A Closed QN

- Models the hardware infrastructure (processors, storage devices, load balancers, etc.) on top of which the software runs.



Mapping the SQN to the HQN

- Mapping the service demand at each SW tier to the HW it uses.
- Example:



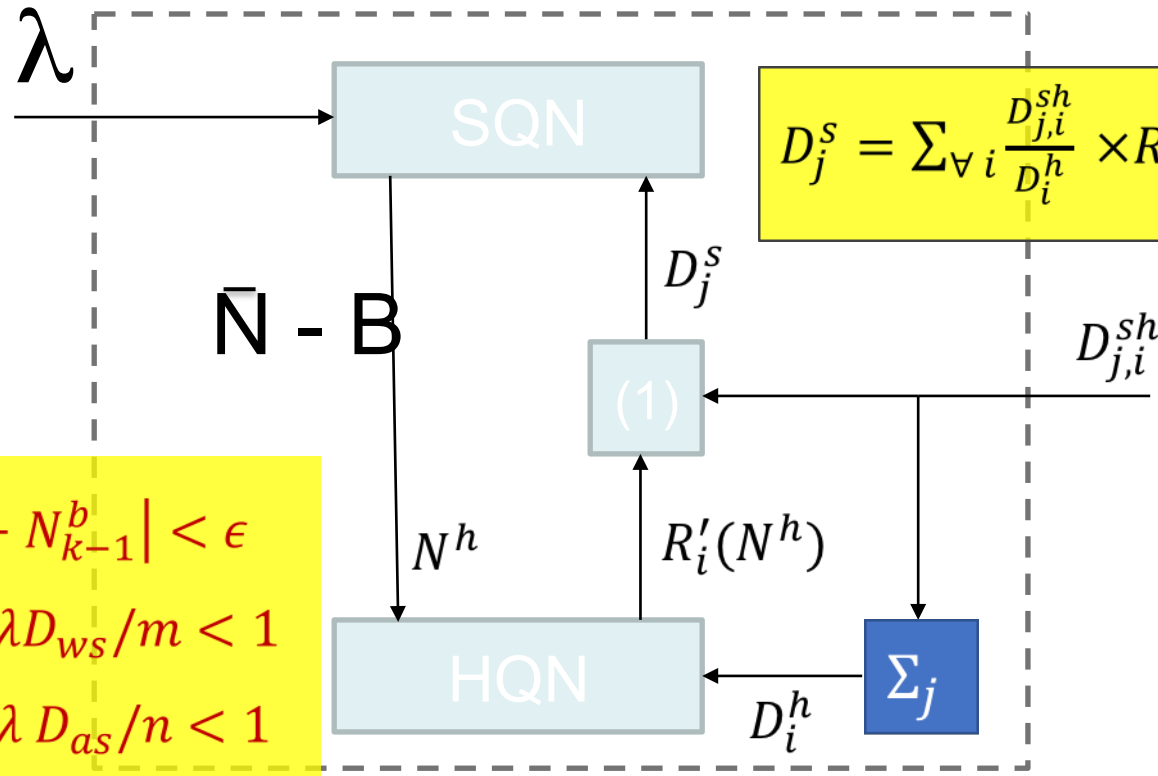
Mapping the SQN into the HQN

- Mapping the service demand at each SW tier to the HW it uses.
- Example:

Hardware Component	Software server			HQN Demands (D_i^h)
	WS	AS	DS	
CPU 1	$f_1\ D_{ws}$	$f_2 D_{as}$	0.00	$f_1\ D_{ws} + f_2 D_{as}$
Disk 1.1	$(1-f_1)D_{ws}$	0.00	0.00	$(1-f_1)D_{ws}$
Disk 1.2	0.00	$(1-f_2)D_{as}$	0.00	$(1-f_2)D_{as}$
CPU 2	0.00	0.00	$f_3 D_{ds}$	$f_3 D_{ds}$
Disk 2.1	0.00	0.00	$(1-f_3)D_{ds}$	$(1-f_3)D_{ds}$
SQN Demands (D_j^s)	D_{ws}	D_{as}	D_{ds}	$D_{ws} + D_{as} + D_{ds}$

- f1, f2 and f3 ∈ [0,1] indicate the fraction of the demand of a software module spent at CPU.

The SQN-HQN Model



$$D_j^s = \sum_{\forall i} \frac{D_{j,i}^{sh}}{D_i^h} \times R_i'(N^h) \qquad (1)$$

$$while \left|N_k^b - N_{k-1}^b\right| < \epsilon$$

$$AND\ U_{ws} = \lambda D_{ws}/m < 1$$

$$AND\ U_{as} = \lambda\, D_{as}/n < 1$$

$$AND\ U_{ds} = \lambda\, D_{ds}/p < 1$$

Modeling the Mulitiered System using SQN-HQN

- $N^b = N^b_{ws} + N^b_{as} + N^b_{ds}$
- $N^b_{ws} = avg\ number\ in\ the\ queue\ - utilization$

$$= \frac{\lambda D_{ws}/m}{1 - \lambda D_{ws}/m} - \frac{\lambda D_{ws}}{m}$$

$$= \frac{(\lambda D_{ws}/m)^2}{1 - \lambda D_{ws}/m}$$

Modeling the Mulitiered System using SQN-HQN

- $N^b = N^b_{ws} + N^b_{as} + N^b_{ds}$
- $N^b_{ws} = \frac{(\lambda D_{ws}/m)^2}{1-\lambda D_{ws}/m}$
- $N^b_{as} = \frac{(\lambda D_{as}/n)^2}{1-\lambda D_{as}/n}$
- $N^b_{ds} = \frac{(\lambda D_{ds}/p)^2}{1-\lambda D_{ds}/p}$

Example – used parameters

$$\lambda = 2.5\ tps$$

$$\text{number of threads in each tier} \begin{cases} m = 10 \\ n = 20 \\ p = 25 \end{cases}$$

$$\text{Demands} \begin{cases} D_{ws} = 0.340\ sec & f_1 = 0.60 \\ D_{as} = 0.200\ sec & f_2 = 0.40 \\ D_{ds} = 0.300\ sec & f_3 = 0.75 \end{cases}$$

Example

- First iterations of the SQN-HQN method -

Iter. No.	D_{ws}	D_{as}	D_{ds}	N^b_{ws}	N^b_{as}	N^b_{ds}	N^b	Abs. Error (Δ)	R_{SQN}
0	0.340	0.200	0.300	0.008	0.001	0.001	0.009	1.000	0.844
1	0.446	0.250	0.374	0.014	0.001	0.001	0.016	0.424	1.076
2	0.505	0.277	0.413	0.018	0.001	0.002	0.021	0.227	1.203
3	0.540	0.292	0.433	0.021	0.001	0.002	0.024	0.129	1.275
4	0.559	0.300	0.445	0.023	0.001	0.002	0.026	0.071	1.316
5	0.571	0.305	0.452	0.024	0.002	0.002	0.027	0.040	1.339
6	0.577	0.308	0.456	0.024	0.002	0.002	0.028	0.023	1.352
7	0.580	0.310	0.458	0.025	0.002	0.002	0.028	0.013	1.359
8	0.582	0.310	0.459	0.025	0.002	0.002	0.029	0.007	1.363

$$\Delta = \left| \frac{N^b(k) - N^b(k-1)}{N^b(k)} \right| .$$

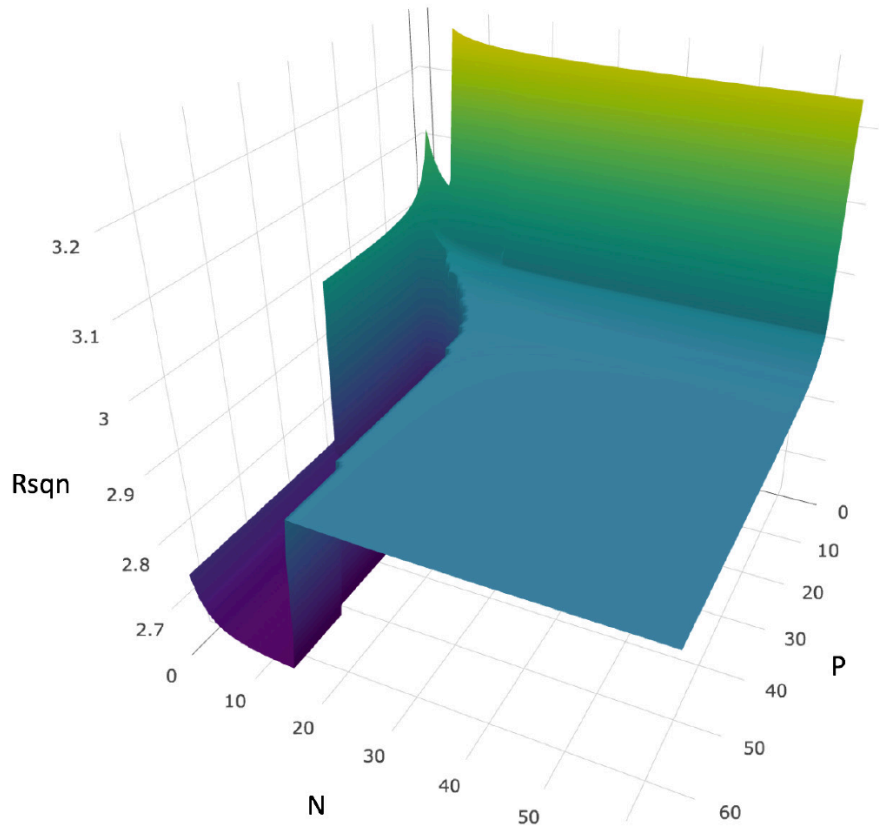
Example

- First iterations of the SQN-HQN method - the HQN:

Iter. No.	R'_{CPU1}	$R'_{\text{Disk1.1}}$	$R'_{\text{Disk1.2}}$	R'_{CPU2}	$R'_{\text{Disk2.1}}$
0	0.401	0.158	0.137	0.293	0.081
1	0.469	0.168	0.145	0.329	0.084
2	0.511	0.173	0.148	0.348	0.085
3	0.534	0.176	0.150	0.360	0.086
4	0.547	0.177	0.151	0.366	0.086
5	0.555	0.178	0.152	0.369	0.086
6	0.559	0.179	0.152	0.371	0.086
7	0.562	0.179	0.152	0.372	0.086
8	0.563	0.179	0.152	0.373	0.086

Example

- R_{SQN} as a function of n and p
- m=30
- λ= 3.6 tps



Near Optimal Number of Threads

- Problem: what is the optimal number of web servers, application servers and database servers that minimizes the response time?
- More precisely, find

$$(m^*, n^*, p^*) = \text{argmin}_{\forall\,(m,n,p)}\{R_{SQN}(m,n,p)\}$$

- an exact optimization solver is not an option because there is no closed form expression for R_{SQN} . Instead, there is an iterative algorithm.

Near Optimal Number of Threads

- A combinatorial search technique can be used to search the solution space for the *near optimal* tuple.
- We used hill-climbing, which starts from an arbitrary point in the search space, and builds a relatively small neighborhood, $\mathcal{N}$, by perturbing the values of each element of (m, n, p)

$$\mathcal{N} = \{(m', n', p') = (m + \delta, n + \delta, p + \delta) \mid$$

- For example,

$$\delta \in \{-1, 0, 1\}, \qquad (\mathrm{m}', \mathrm{n}', \mathrm{p}') \neq (m, n, p),$$

$$\frac{\lambda D_{ws}}{m'} < 1, \qquad \frac{\lambda D_{as}}{n'} < 1, \qquad \frac{\lambda D_{ds}}{p'} < 1 \; and$$

$$m' \geq 1, n' \geq 1, p' \geq 1\}$$

Near Optimal Number of Threads

- Additionally, the number of threads of each type in each point in the neighborhood also has to satisfy the memory constraint in each physical machine.
- In other words,

$$\begin{aligned} m \times M_{ws} + n \times M_{as} &\leq \mathrm{AM_{M1}} \\ p \times M_{ds} &\leq \mathrm{AM_{M2}} \end{aligned}$$

where M_{ws}, M_{as} and M_{ds} are the memory footprint for each thread type, and AM_{M1} and AM_{M2} stand for the available memory on machines 1 and 2.

Hill Climbing Algorithm

Algorithm 1 Hill-climbing algorithm

```
(m, n, p) ← StartConfig;
MinRespTime ← RespTime ((m, n, p));
for NumRestarts = 1 to MaxRestarts do
  NumIter ← 1;
  /* Do one start/restart */
  while NumIter ≤ MaxIter do
    /* Build neighborhood of (m, n, p). See Eq. 10. */
    N ← Neighborhood ((m, n, p));
    /* Find config. with smallest resp. time in N */
    (m', n', p') ← argmin_{(m,n,p)∈N} RespTime((m, n, p))
    /* Find resp. time of (m', n', p'). See Section IV */
    NewRespT ← RespTime((m', n', p'));
    if NewRespT < MinRespTime then
      /* Resp. time of (m', n', p') < that of (m, n, p) */
      MinRespTime ← NewRespT;
      /* move the neighborhood center to (m', n', p') */
      (m, n, p) ← (m', n', p');
      /* increment number of iterations */
      NumIter ← NumIter + 1
    else
      /* No point in N has a smaller resp. time */
      /* End of this iteration */
      NumIter ← MaxIter + 1;
    end if
  end while
  /* Record local optimum in LocalOpt array */
  LocalOpt[NumRestarts].Config ← (m, n, p);
  LocalOpt[NumRestarts].RespT ← MinRespTime;
  /* set a new point for a random restart */
  (m, n, p) ← RandomConfig;
end for
/* Find index of best local optimum */
OptConfigIndex ← argmin_i LocalOpt[i].RespT;
/* Return best local optimum configuration */
Return (LocalOpt[OptConfigIndex].Config)
```

Near Optimal Results for Varying Arrival Rates – used parameters

$$\text{Demands} \begin{cases} D_{ws} = 0.340\ sec & f_1 = 0.60 \\ D_{as} = 0.200\ sec & f_2 = 0.40 \\ D_{ds} = 0.300\ sec & f_3 = 0.75 \end{cases}$$

$$\text{Memory footprint} \begin{cases} M_{ws} = 100\ MB \\ M_{as} = 150\ MB \\ M_{ds} = 100\ MB \end{cases}$$

$$\text{Available memory} \begin{cases} AM_{machine1} = 8000\ MB \\ AM_{machine2} = 4000\ MB \end{cases}$$

$$\text{Hill climbing} \begin{cases} MaxIter = 200 \\ MaxRestarts = 50 \end{cases}$$

Near Optimal Results for Varying Arrival Rates

Arrival rate (tps)	m	n	p	R_{SQN}
2.0	41	26	39	1.084
2.5	44	24	39	1.355
3.0	44	24	39	1.915
3.5	78	1	39	1.307
4.0	2	52	39	1.202
4.5	44	24	2	1.152
5.0	3	51	39	0.989
5.5	4	50	39	0.915
6.0	4	50	39	0.929
6.5	5	50	39	0.894
7.0	6	49	39	0.878

Concluding Remarks

- Simple approach.
- Open, closed, and multiclass QNs can be used at the SQN.
- SQNs can include non-software resources that are not mapped to hardware resources.
- HQNs are closed and can be multiclass.
- Any technique can be used to solve the SQN and HQN. This includes any known approximation to multiple-server devices, priorities, simultaneous resource possession, etc.