Routing Mechanism for Static Load Balancing in a Partitioned Computer System with a Fully Connected Network

Hitoshi Oi [1] and Bing-rung Tsai [2]

[1] Dept. of Computer Science & Engineering
Florida Atlantic University
Boca Raton, FL 33431
hitoshi@cse.fau.edu
[2] Kuokoa Networks, Inc.
Santa Clara, CA 95054
btsai@kuokoa.com

Abstract. System partitioning provides the users of high-performance parallel servers with the flexibility in resource allocation and dynamic reconfiguration as well as fault isolation. However, the bandwidth of links that connect different domains can be wasted while links within the same domains are congested. In this paper, we present a routing mechanism that can utilize the bandwidth of otherwise unused links to balance the message traffic and lead to lower message latencies for the latency-sensitive transactions. The performance of the proposed routing mechanism was studied using an analytical model with on-line transaction processing type workload parameters. The results indicated the proposed routing mechanism reduced the congestion on the direct paths significantly and lowered the queuing delay for the links. For example, when a 4-cluster system with a fully connected network with the bandwidth of 3.2GB/s per link is partitioned into two 2-cluster domains, the queuing delay was reduced from 53ns to 37ns and resulted in the improvement of CPI by 2%.
Keywords: System partitioning, distributed shared memory, interconnection network, message routing.

1 Introduction

Cache coherent non-uniform memory architecture (CC-NUMA) has the advantages of both the shared memory space of SMPs and the scalability of loosely-coupled multi-computers. Till mid-90s, CC-NUMA multiprocessors were mainly used for engineering or scientific applications. Nowadays, they are used for running different commercial applications, such as on-line transaction processing (OLTP), decision support systems (DSS), or web-servers, various operating systems. Each system runs different applications time to time, and the required system resources, such as processors, memory, storage devices, and network, also vary.

System partitioning enables users to divide a system into several domains. Each domain behaves as a separate system, where users can run different applications with a different operating system. The resource allocation for each domain can be changed without shutting down the system or physical re-configurations (such as reconnecting cables) [1, 2, 3, 4]. In a CC-NUMA multiprocessor, memory modules are physically distributed among nodes and cache coherence is maintained by sending messages between nodes over the interconnect network. When a CC-NUMA multiprocessor is partitioned into domains, message traffic are routed through the links that connect nodes belonging to the same domain. Thus, the bandwidth of the other links (those connecting nodes belonging to different domains) are unused.

In this paper, we present a static routing mechanism that can utilize the bandwidth of such unused links to balance the message traffic and lead to lower message latencies. The proposed routing mechanism classifies message traffic into two categories, latency sensitive and non-latency sensitive transactions. The latter transactions are routed through the indirect paths to balance the message traffic on the direct and indirect paths.

This paper is organized as follows. In Section 2, the proposed routing mechanism is described. In Section 3, the performance advantage of the proposed routing mechanism is evaluated. Section 4 discusses related work and the conclusion is provided in Section 5.

2 Routing Mechanism

In this section, we describe the proposed routing mechanism and its operations for the different transaction categories. A fully interconnected three cluster system is shown in Fig. 1. Each cluster has a CPU node, an I/O node and a router. From a router, there are two links that connect the cluster to others. Thus, in this example, the router is a 4×4 crossbar switch. Small numbers inside the square of the router indicate the ports to which nodes and links are connected. Note that this configuration is simplified in terms of number of nodes and hierarchical level for easier explanation. In Section 3, we will use more realistic configurations.

Consider a case where the system in Fig. 1 is partitioned into two domains: Domain 0 that consists of clusters A and B, and Domain 1 that consists of Cluster C alone. Each domain behaves as an independent CC-NUMA system and has its own address space which is only accessible to the nodes within the domain. Thus, the link between Clusters A and B (the link drawn with a bold line in Fig. 1) is heavily used while other links are not used at all. In the proposed routing mechanism, these unused links are utilized to balance the message traffic in the system.

In the proposed scheme, all the transactions are classified into two categories: either I/O transactions or processor-memory transactions (or non-I/O transactions for short). I/O transactions are less sensitive to the latency. For example, when a file is opened for an application, direct memory access (DMA) trans-

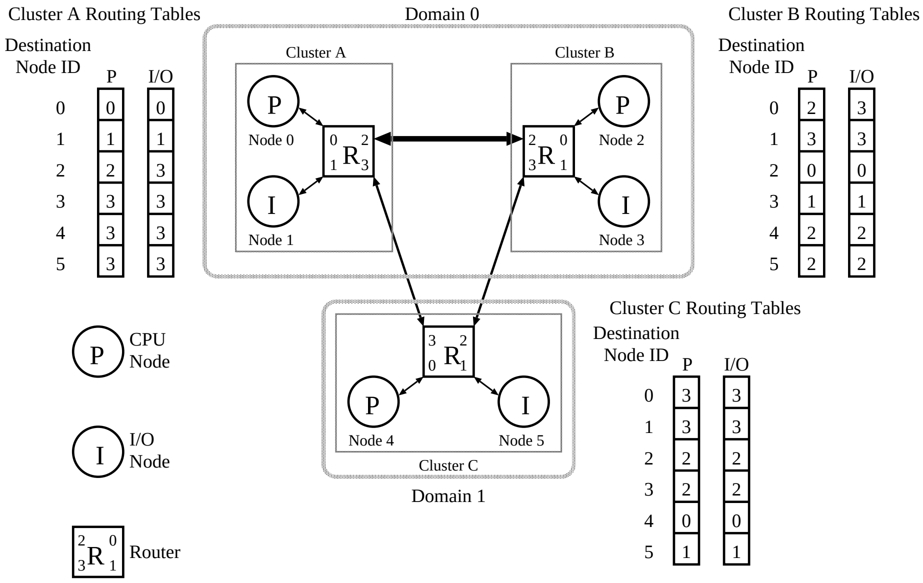

Fig. 1. A Fully Interconnected 3 Cluster System Partitioned into Two Domains and Routing Tables

actions are initiated by the I/O node so that future accesses to the file by the application will hit at the memory buffer. On the other hand, cache fill transactions initiated by cache misses of processors are latency sensitive. The processor could be blocked until the cache fill transaction is completed [3]. By routing I/O transactions through the unused links, it is possible to reduce traffic on the links that connects cluster within the same domain and this leads to a lower latency of non-I/O (i. e. processors-memory) transactions that are latency-critical.

In this mechanism, each cluster has two routing tables, one for I/O and another for non-I/O transactions. In a message packet, there is an I/O bit indicating whether the originator of the message is an I/O node in the cluster or not. This I/O bit is used at the router to choose one of routing tables. Each node in the system (either CPU or I/O node) has a unique Node ID as shown in the table in Fig. 1 and entries in the routing tables are indexed by the Node ID.

Below, operations of the proposed routing mechanism are described with four different transactions: memory access from a processor and its response, and I/O access from a processor and its response.

---

[3] modern microprocessors employ techniques to overlap cache-fill and other operations, such as nonblocking cache, or multi-threading. However, blocking still occurs if the processor reaches to the point where it uses the data in the missed cache block

2.1 Processor-Memory Transactions

A processor in Node 0 accesses a memory module in Node 2 (Processor-Memory Transaction in Fig. 2). First, the processor sends a request message to Node 2:

1. Node 0 sends a request message with $I/O = 0$.
2. Since $I/O = 0$, Non-I/O Routing Table (labeled "P" in the table in Fig. 1) is used at the router in Cluster A. The entry corresponding to Node 2 (destination of the request message) indicates that the message is to be routed to the Port 2 of the router (to the cable directly connected to Cluster B).
3. The message is transmitted over the cable and reaches at the router in Cluster B. Since this message is not originated from an I/O node in the cluster, its $I/O = 0$. Thus, Non-I/O Routing Table is used.
4. Non-I/O table indicates that the message is to be routed to Port 0, which is the destination of the message (Node 2).

After the memory access is completed, Node 2 sends a response message back to the processor in Node 0.

1. Node 2 sends a response message with $I/O = 0$.
2. Since $I/O = 0$, Non-I/O Routing Table is used at the router in Cluster B. The entry corresponding to Node 0 (destination of the response message) indicates that the message is to be routed to the Port 2 of the router (to the cable directly connected to Cluster A).
3. The message is transmitted over the cable and reaches at the router in Cluster A. Since this message is not originated from an I/O node in the cluster, its $I/O = 0$. Thus, Non-I/O Routing Table is used.
4. Non-I/O table indicates that the message is to be routed to Port 0, which is the destination of the response message (Node 2).

2.2 Processor-I/O Transactions

A processor in Node 0 accesses an I/O device in Node 3 (I/O Transaction in Fig. 2). Similar to the previous example, the processor first sends a request message to Node 3:

1. Node 0 sends a request message with $I/O = 0 sinceitisaCPUnode$.
2. Since $I/O = 0$, Non-I/O Routing Table is used at the router in Cluster A. The entry corresponding to Node 3 (destination of the request message) indicates that the message is to be routed to the Port 3 of the router (to the cable connected to Cluster C).
3. The message is transmitted over the cable and reaches at the router in Cluster C. Since this message is not originated from an I/O node in the cluster, its $I/O = 0$. Thus, Non-I/O Routing Table is used.
4. Non-I/O table indicates that the message is to be routed to Port 2, to the cable connected to Cluster C

5. The message is transmitted over the cable and reaches at the router in Cluster B. Since this message is not originated from an I/O node in the cluster, its $I/O = 0$. Thus, Non-I/O Routing Table is used.
6. Non-I/O table indicates that the message is to be routed to Port 1, which is the destination of the request message (Node 3).

After the I/O access is completed, Node 3 sends a response message back to the processor in Node 0.

1. Node 3 sends a response message with $I/O = 1$ (since Node 3 is an I/O node).
2. Since $I/O = 1$, I/O Routing Table (labeled "I/O" in the table in Fig. 1) is used at the router in Cluster B. The entry corresponding to Node 0 (destination of the response message) indicates that the message is to be routed to the Port 3 of the router (to the cable connected to Cluster C).
3. The message is transmitted over the cable and reaches at the router in Cluster C. Since this message is not originated from an I/O node in the cluster, its $I/O = 0$. Thus, Non-I/O Routing Table is used.
4. Non-I/O table indicates that the message is routed to Port 3, to the cable connected to Cluster A.
5. The message is transmitted over the cable and reaches at the router in Cluster A. Since this message is not originated from an I/O node in the cluster, its $I/O = 0$. Thus, Non-I/O Routing Table is used.
6. Non-I/O table indicates that the message is routed to Port 0, which is the destination of the response message (Node 0).

3 Performance Evaluation

In this section, we study the effectiveness of the proposed routing mechanism using an analytical model. First, we describe the target system configurations. Then the methodology of the evaluation and the results follow.

3.1 Architecture Model

We assume hierarchical CC-NUMA system configurations that can be partitioned into multiple domains. The system consists of four of clusters and clusters are fully connected by the point-to-point links. A cluster consists of four SMP nodes, two I/O nodes, a directory controller, a crossbar switch that connects the nodes within the cluster as well as the inter-cluster network. An SMP node has four processors, a memory controller and memory modules (Fig. 3).

We evaluate the performance of the proposed routing mechanism in two configurations in Fig. 4: there are four clusters in the system and they are partitioned into two domains. In Fig. 4 (a), there are two domains: one consisting of clusters 0 and 1 and another of clusters 2 and 3 Links between clusters 0 and 1

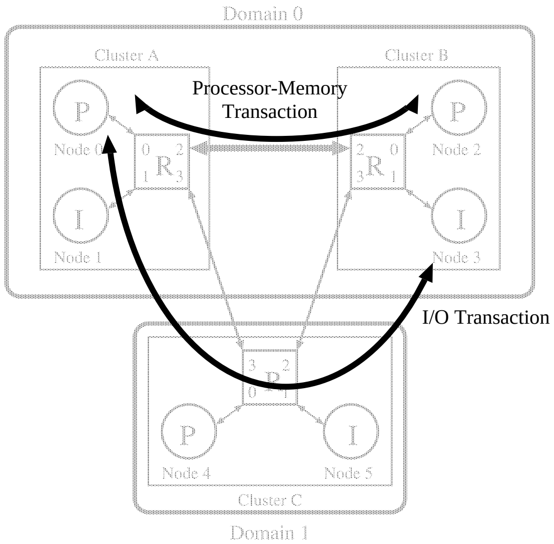

Fig. 2. Message Flow for Processor-Memory and I/O Transactions

and clusters 2 and 3 (drawn with bold lines in the figure) are used for processor-memory transactions and other links are used for I/O transaction. Similarly, in Fig. 4 (b), clusters 0 to 2 form a domain while cluster 3 is used as a single cluster domain. Table 1 lists key system parameters, which are chosen in consideration of the current implementation technologies. Memory access latencies are for read accesses to unmodified memory blocks. Using an analytical model, which is described in the next section, the utilization and queuing delay of the links connecting clusters and clock per instruction (CPI) are derived and used as performance indices.

Parameter	Value
Proc/SMP Node	4
Proc Clock	1.1GHz
Bus Clock	200MHz
Local Memory	250ns
Remote Memory (same cluster)	385ns
Remote Memory (other cluster)	550ns
Link Bandwidth	3.2, 3.6, 4GB/s

Table 1. Key System Parameters

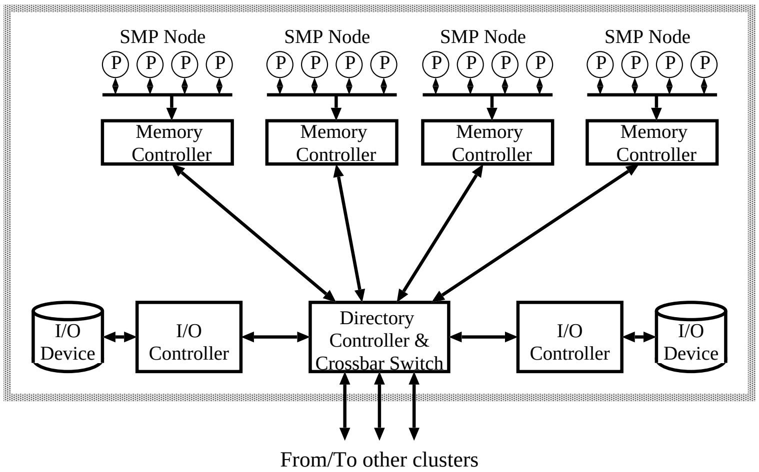

Fig. 3. Cluster Architecture

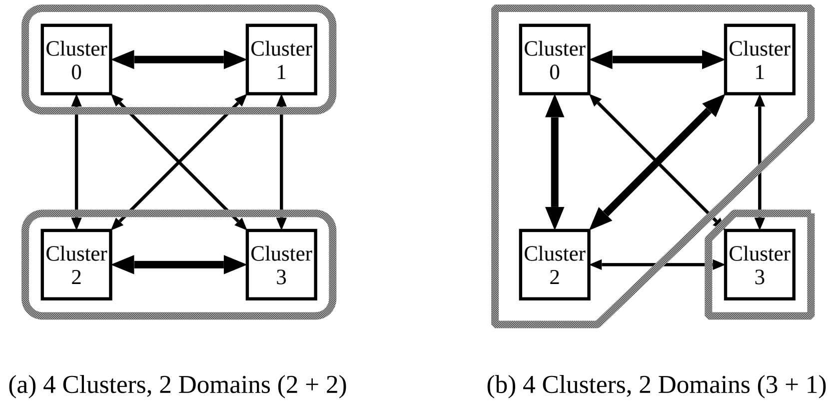

Fig. 4. System Configurations

3.2 Performance Model

We developed an analytical model based on the one used in our previous project at HAL Computer Systems [5].

The primary index of the performance is clock per instruction, CPI which is obtained as follows:

$$CPI = CPI_{perfectL2} + f_b \times mpi \times cpm \tag{1}$$

where,

$CPI_{perfectL2}$ is the processor CPI with perfect (no miss) L2 cache,
f_b is a *blocking factor*, which reflects the effectiveness of latency hiding techniques in both hardware and software,
mpi is the average L2 miss rate (miss per committed instruction),
cpm is the average miss penalty in clock cycles.

To determine the value of cpm, L2 miss is classified into various transaction scenarios by the access mode (read/write), state of the main memory, state of other L2 caches, locations of the accessed block's home and the sharer or owner L2 cache. Each transaction provides the usage of resources in the system, such as memory controller, directory controller, shared bus in the SMP node, links between clusters.

Next, we derive cpm in (1), which is given by

$$cpm = cpm_{NoContention} + QueingDelay \tag{2}$$

where $cpm_{NoContention}$ is the L2 cache miss penalty when there is no contention on the system resources, while $QueingDelay$ is the delay caused by the contention on the system resources. Note that $QueingDelay$ is a function of CPI: the lower CPI, the more frequently resources are used and higher contention. Thus

$$CPI = CPI_{perfectL2} + f_b \times mpi \times cpm(CPI). \tag{3}$$

The queuing delay of each resource is approximated by M/D/1 formula. The probability of each transaction scenario was chosen from our experiences in the past projects together with an OLTP-type workload [6, 7]. Also, the following assumption are used:

- Memory modules and I/O devices are uniformly distributed.
- The ratio of read and write access is $2:1$.
- DMA I/O traffic is assumed $100MB/(20000Transaction/min)$

With these workload assumptions, the derivation of (3) is iterated until it is converged.

3.3 Results

Configuration (a) in Fig. 4 is a 4-cluster system partitioned into two 2-cluster domains . Table 2 shows the link utilization and queuing delay on the direct paths (drawn with bold lines in Fig. 4) and CPI, derived from the analytical model. In the table, *DP* columns are the values when only the links on the direct paths are used while *Prop* are for the proposed scheme. The proposed routing scheme reduced the link utilization significantly and resulted in the lower queuing delay. At the link bandwidth of 3.2GB/s, the queuing delay was reduced by 30% and the CPI was improved by 2%. At the higher bandwidth values, 3.6GB/s and 4.0GB/s, the queuing delay was reduced by 25% and 24%, and CPI was improved by 1% and 0.5%, respectively.

	Link Bandwidth					
	3.2GB/S		3.6GB/S		4.0GB/S	
	DP	Prop	DP	Prop	DP	Prop
Link Utilization	82.8%	77.7%	76.2%	71.0%	68.7%	63.7%
Queuing Delay	53.0ns	36.8ns	31.8ns	23.3ns	19.4ns	14.8ns
CPI	4.155	4.083	4.061	4.023	4.006	3.986

Table 2. Configuration (a) Result

	Link Bandwidth					
	3.2GB/S		3.6GB/S		4.0GB/S	
	DP	Prop	DP	Prop	DP	Prop
Link Utilization	*sat*	*sat*	*sat*	84.1%	81.9%	76.9%
Queuing Delay	N/A	N/A	N/A	45.3ns	35.8ns	25.4ns
CPI	N/A	N/A	N/A	4.760	4.696	4.626

Table 3. Configuration (b) Result

Configuration (b) in Fig. 4 is a 4-cluster system partitioned into two (3 and 1-cluster) domains. Table 3 shows the link utilization, queuing delay, and CPI values of the 3-cluster domain derived from the analytical model. The link bandwidth of 3.2GB/S was not sufficient for the high transaction rate assumed in the workload. The link was saturated (link utilization exceeded 100% in the analytical model) in both *DP* and *Prop*. In these cases, the system behavior is non-linear and the performance cannot be predicted by the analytical mode. At the bandwidth of 3.6GB/s, however, *DP* and *Prop* behaved differently. with *DP* the links are still saturated while with *Prop* the link utilization is far below saturation. At the bandwidth of 4.0GB/s, the queuing delay was reduced by 30% and CPI was improved by 1.5%.

4 Related Work

Modern high-performance NUMA systems with system partitioning feature include HP Superdome [1], NEC AZUSA [2], Compaq AlphaServer [3], Unisys Enterprise Server ES 7000 [4]. In particular, HP Superdome has a similar structure as the architectural mode we assumed in this paper: a number of processors form a cell, and cells are fully connected by the network of crossbar switches. However, as to the best of our knowledge, none of the existing systems including above can utilize the inter-domain links to alleviate the traffic on the intra-domain links.

In addition to the normal interconnection network for the inter-processor communications, NUMA-Q of IBM [8] has a dedicated network of fiber channel switches for connecting I/O devices and processor nodes, called Multipath I/O. In principle, Multipath I/O and the proposed routing mechanism are the same approach in the sense that both try to reduce the traffic on the inter-processor network by routing I/O traffic to alternative paths. The difference is in the cost-effectiveness: while NUMA-Q invested in the extra hardware (fiber channel switches) for the I/O traffic, the proposed scheme utilizes the bandwidth of unused links resulted from the system partitioning.

In a parallel computer system with the message-passing communication model, which has a much higher inter-processor communication latency than the shared memory model, there are many sophisticated dynamic (adaptive) routing algorithms [9]. They are, however, rarely used for the interconnection network of NUMA systems. Sophisticated adaptive routing algorithms inherently increase the routing latencies. This conflicts with one of the objectives in the interconnection network of NUMA systems, which is to reduce the remote access latency.

In this paper, we assumed the indirect paths were to be used for I/O transactions, which are normally implemented as uncached DMA operations [10]. If the proposed routing scheme is used for cache coherent memory transactions, the race conditions caused by the multiple paths between nodes must be solve [11].

5 Conclusion

Most of current commercial parallel servers have cluster structures and they can be partitioned into several domains to better utilize the system resources to fit users need as well as fault isolation. In this paper, we have presented a static routing scheme that balances the message traffic in a partitioned system. To minimize the latency of remote memory accesses by processors, it is preferable to route messages over the links that directly connect clusters in the same domain. However, this routing scheme leads to an unbalance of the message traffic: the links connecting clusters in the same domain are heavily congested while other links are not used. As an initial study, we have evaluated the performance of the proposed scheme with an analytical model using workload parameters that modeled an OLTP-type application.

In addition to balancing the message traffic, the proposed scheme has several advantages. The hardware resource required for its implementation is small and complexity is fairly low. Thus, it is expected that the effect on the latency for the transaction on the direct paths is minimum. Another advantage is, since the proposed scheme is static, it does not require to keep track of strayed transactions, which is a common problem in more sophisticated dynamically adaptive routing mechanisms.

The proposed routing scheme can be applied to a system in which each cluster (in the terminology used in this paper) is fully connected to every other cluster with a dedicated link. For example, Superdome of Hewlett Packet [1] is met with this structural requirement and the proposed scheme can be applied to Superdome.

The topics of further research include studying dynamic network behavior using execution driven simulations or other methods, use of more various workload such as web-servers or decision support systems (DSS). In this paper, I/O transactions represented latency-insensitive messages. The proposed scheme can be used for other types of transactions, such as prefetching memory blocks into processors' caches, writing back replaced cache blocks to the main memory, and the effectiveness of the proposed scheme for these transaction should also be studied.

Acknowledgment

The proposed routing mechanism was developed in the course of the ASURA project which was conducted at HAL Computer Systems Inc., Campbell, California. The authors would like to appreciate the encouragement, discussion, and valuable comments from the colleagues in the project.

References

1. "HP Virtual Partitions", *a white paper from Hewlett-Packard Company*, September 2000
2. Fumio Aono and Masayuki Kimura, "The Azusa 16-way Itanium Server", *IEEE Micro*, Vol. 20, Issue. 5, 54–60, Sep-Oct 2000.
3. "AlphaServer GS80, GS160, and GS320 Systems Technical Summary", Compaq.
4. "Cellular Multiprocessing Shared Memory", White Paper, Unisys, September 2000.
5. "Distributed Shared Memory System Performance", HAL internal document, February 1998.
6. Russell M. Clapp, "STinG Revisited: Performance of Commercial Database Benchmarks on a CC-NUMA Computer System", in *Proceedings of Fourth Workshop on Computer Architecture Evaluation using Commercial Workloads*, Monterrey, Mexico, January 2001.
7. Don DeSota, "Characterization of I/O for TPC-C and TPC-H workloads", in *Proceedings of Fourth Workshop on Computer Architecture Evaluation using Commercial Workloads*, Monterrey, Mexico, January 2001.
8. "The IBM NUMA-Q enterprise server architecture", IBM, January 2000.
9. M. Boari, A. Corradi, C. Stefanelli and L. Leonardi, "Adaptive Routing for Dynamic Applications in Massively Parallel Architectures", *IEEE Parallel & Distributed Technology: Systems & Applications*, Vol. 3 Issue 1, Spring 1995, pp61–74.
10. D. Lenoski and W.-D. Weber, "Scalable Shared-Memory Multiprocessing", Morgan Kaufmann Publishers, 1995.
11. D. Dai and D. Panda, "Exploiting the benefits multiple-path network in DSM systems: architectural alternatives and performance evaluation", *IEEE Trans. on Computers*, Vol. 48 Issue 2 , Feb. 1999 pp236–244.

This article was processed using the LaTeX macro package with LLNCS style