

Distributed Service Deployment over Programmable Networks

Robert Haas*, **Patrick Droz***, **Burkhard Stiller****,

*IBM Zurich Research Lab,

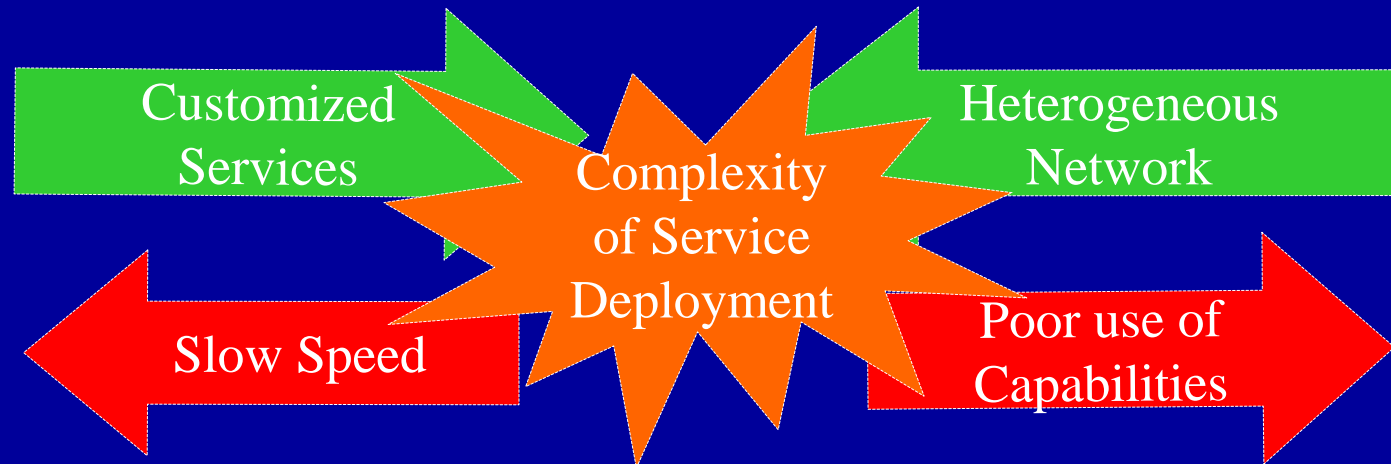
**Computer Engineering and Networks Lab, ETHZ.

DSOM 2001, Nancy, France, October 2001

Content

- Motivation
 - Problem statement
 - Related work
- Service Deployment Categories
- Services and Capabilities Descriptions
- Procedure overview
 - Example of service deployment
- Conclusion

Problem Statement



- Provide automated service deployment
 - Hierarchical for large networks
 - service-generic

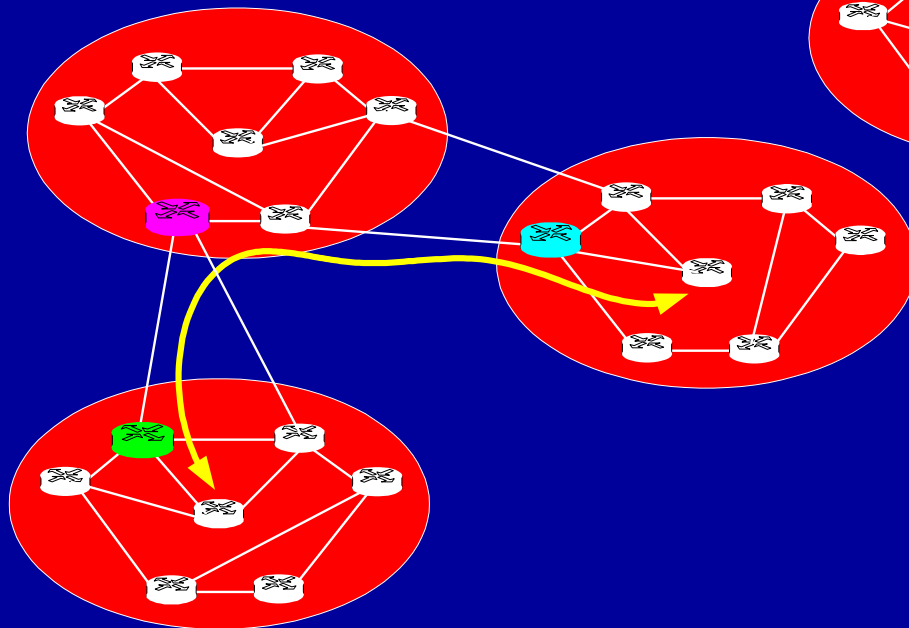
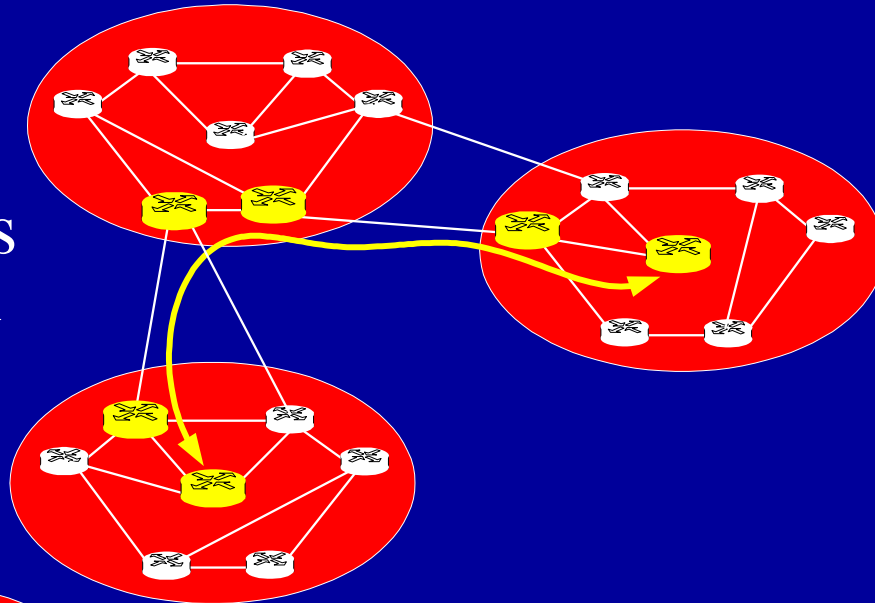
Basic Information Types
Topology
Capabilities
Performance
Cost

Related Work

- in active networks
 - service deployed along the path taken by the capsule
 - service destined for that session
- in mobile agents
 - navigation model
- in programmable networks
 - service loaded on-demand, manual coordination
- in IN (Intelligent Network)
 - service deployed at a central point (SCP)
- in sensor networks

Service-Deployment Categories

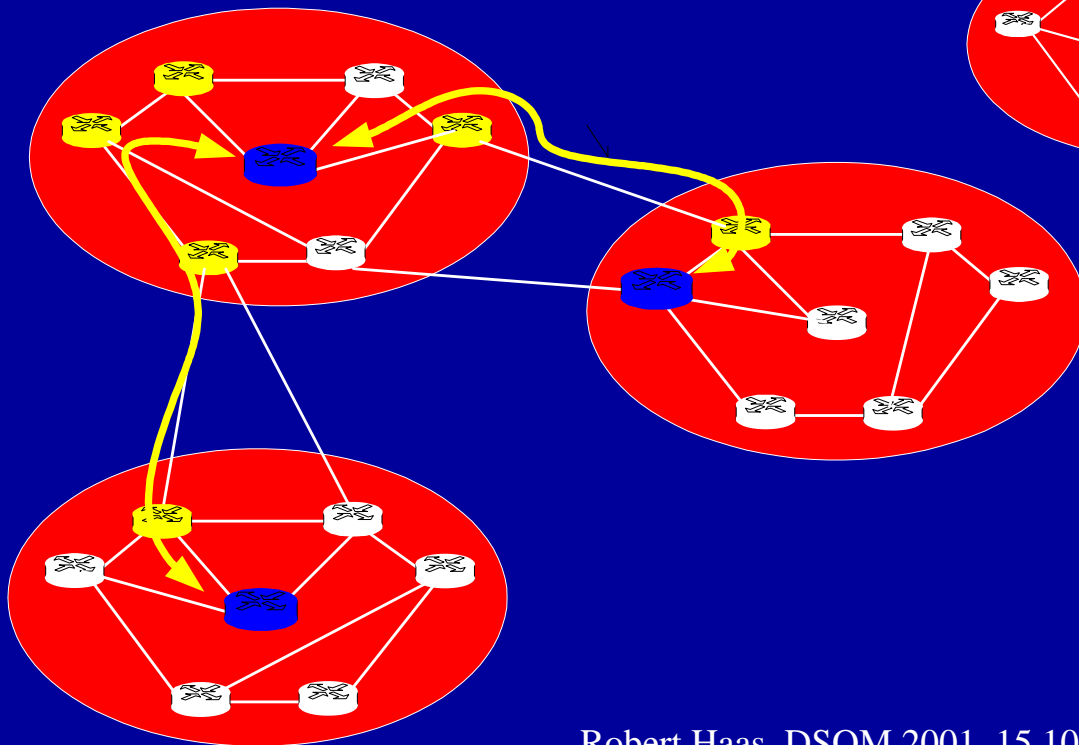
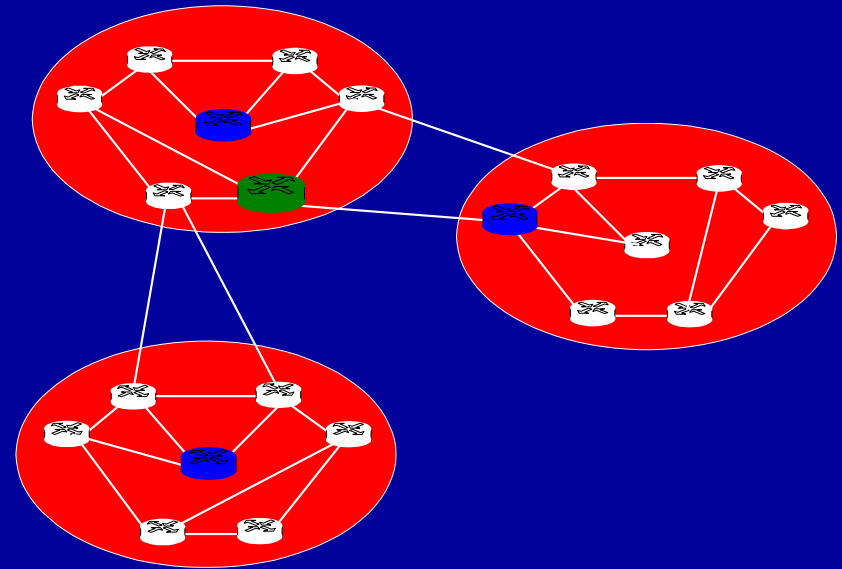
- Path-based
 - Continuous
 - Diff-Serv



- Path-based
 - Sparse
 - Filtering/Transcoding/Compression

Service-Deployment Categories (2)

- Node-Based
 - Web-caching



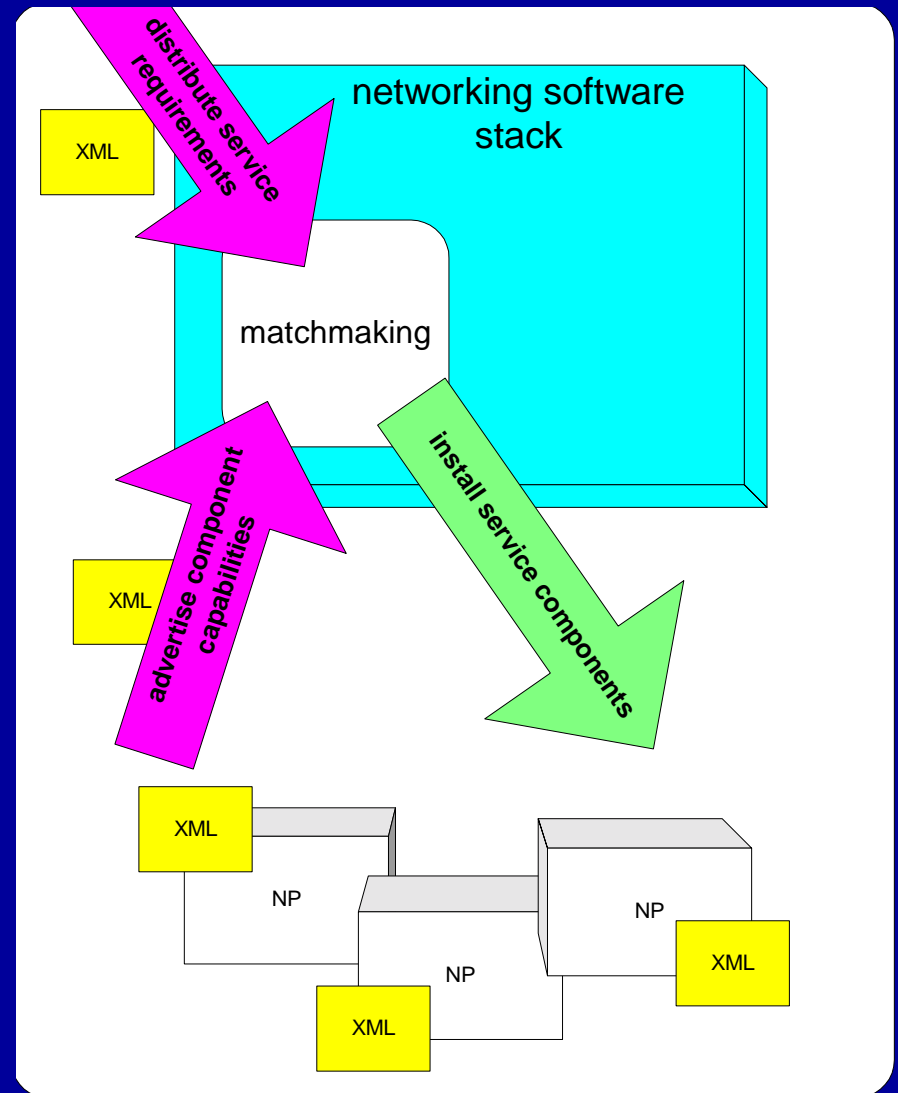
- Path-and-Node-Based
 - VPN

Service and Capabilities Description

- Common description of:
 - Service requirements
 - Components capabilities
- Matchmaking process produces a *metric*

Component Description

- Control & Traffic APIs supported
- Performance
- Capabilities



NP XML Description

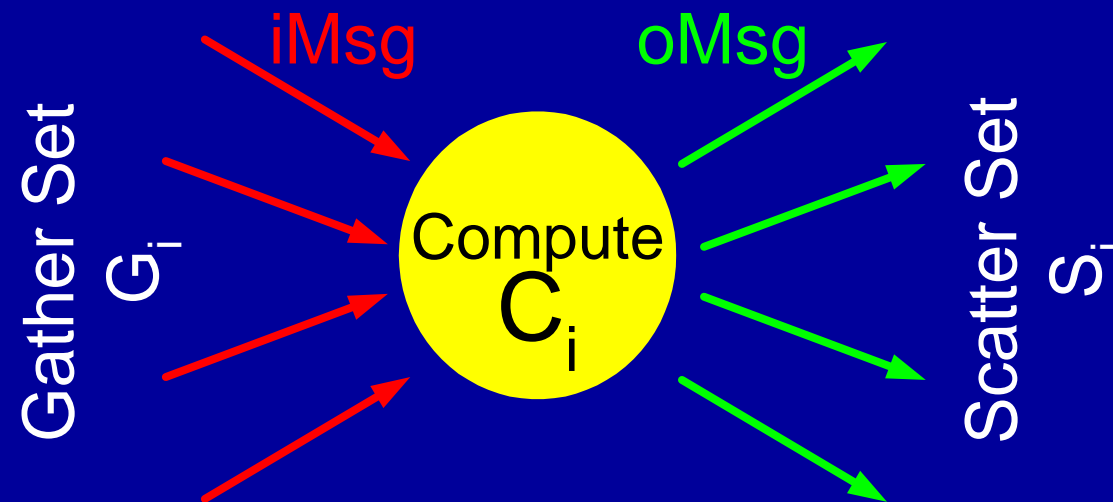
```
<Network_Processor>
  <base_capabilities>
    <API_supported> PIN_1520, MIB </API_supported>
    <PIN_1520_specifics>
      <version> 1.3 </version>
    </PIN_1520_specifics>
    <general>
      <processing>
        <speed> 500 MHz </speed>
      </processing>
      <scheduling>
        <total_bandwidth> 100 Mbit/s </total_bandwidth>
        <type> WFQ </type>
        <max_queues> 1000 </max_queues>
      </scheduling>
      <buffers_management>
        <total_buffer_size> 1 MB </total_buffer_size>
        <max_buffer_pools> 16 </max_buffer_pools>
        <buffer_sharing> yes </buffer_sharing>
        <RED> yes </RED>
      </buffers_management>
    </general>
    <resource_usage>
      // current usage for the defined capabilities
    </resource_usage>
  </base_capabilities>
  <diff_serv> // absent if NP does not provide
              // explicit support for diff-serv
  <API_supported> standard_MIB </API_supported>
  <general>
    <classifier>
      <fields> 6 </fields>
      // etc
    </classifier>
    // etc
  </general>
  <resource_usage>
    // current usage for the defined capabilities
  </resource_usage>
</diff_serv>
// etc
</Network_Processor>
```

Service-Deployment Procedure

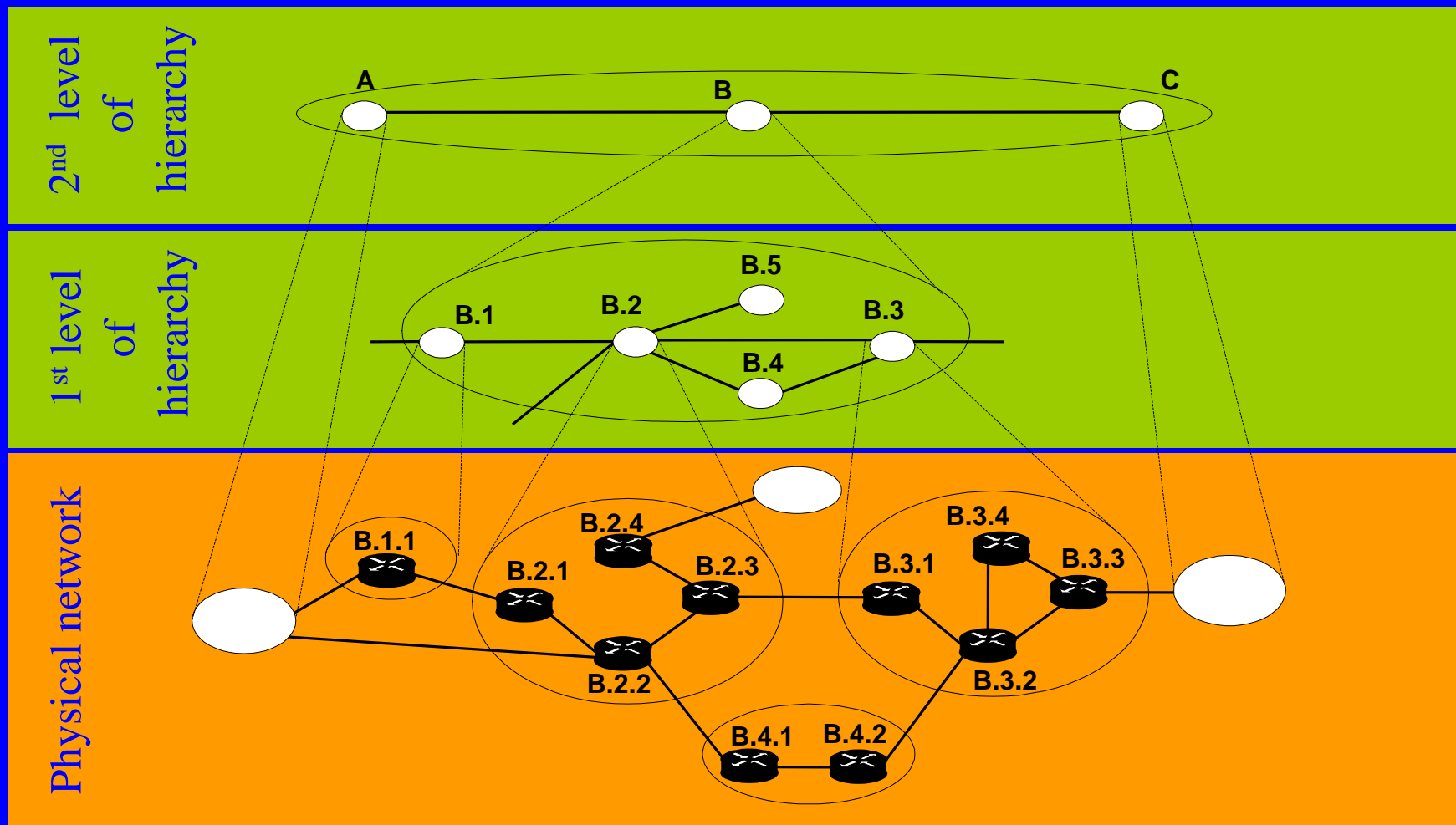
Solicitation	Summarization	Dissemination	Installation	Advertisement
			manual deployment and automatic configuration	
	automatic deployment with generic metrics and automatic configuration			
automatic deployment with custom metrics and automatic configuration				

HIGCS

- Hierarchical Iterative Gather-Compute-Scatter
 - distribute service deployment computations across the hierarchy
- Succession of iterations defined by:



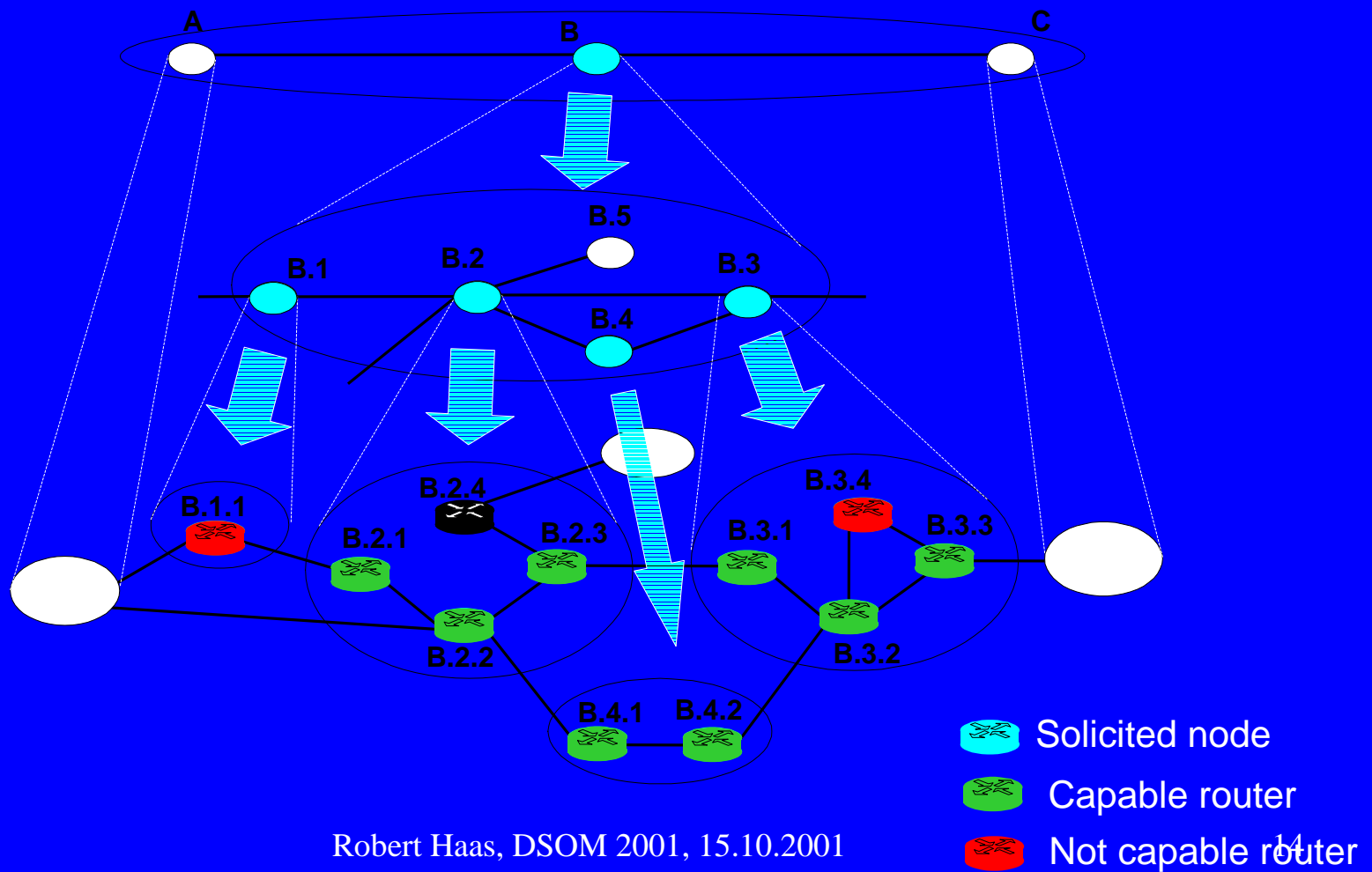
Path-Based Example: Diff-Serv



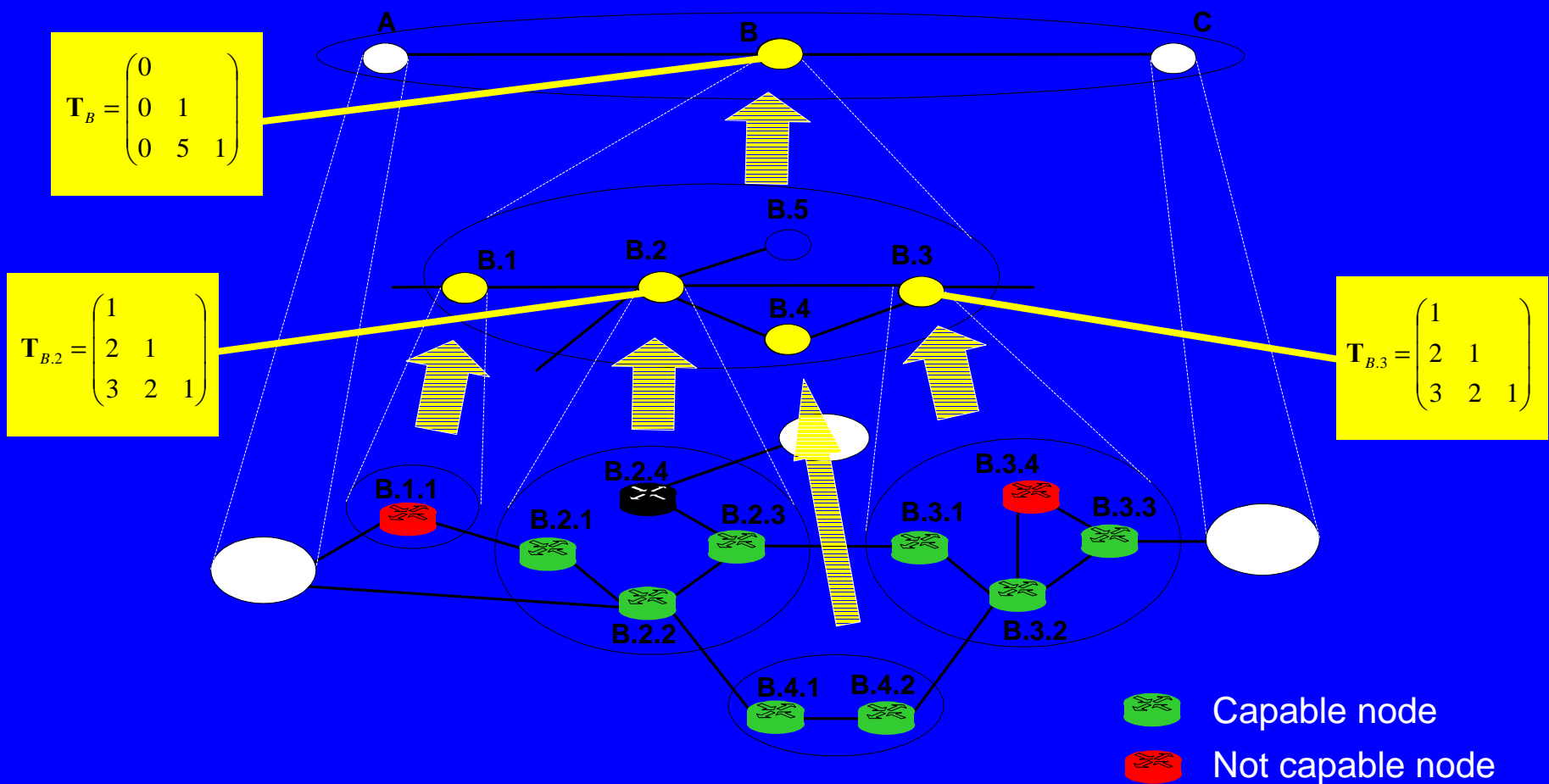
HIGCS (Hierarchical Iterative Gather Compute Scatter)

C_0		DS-solicit
S_0	←	SelectAllNodesBetween($iMsg.ends$)
$oMsg_n.ends$	←	SelectNeighborNodes($n / n \in S_0$)
G_1	←	S_0
C_1		DS-summarize
S_1	←	GetLogicalNode()
$oMsg.sMetric$	←	if IsLogicalNode() then SummarizeMetrics($oMsg_j.sMetric, j \in G_1$) else CreateMetric($iMsg.servSpec$)
G_2	←	S_1
C_2		DS-disseminate
S_2	←	if IsLogicalNode() then SelectNodesOnShortestPath($iMsg.ends$) else null
$oMsg_n.ends$	←	SelectNeighborNodes($n / n \in S_2$)
G_3	←	S_2
C_3		DS-install
S_3	←	GetLogicalNode()
$oMsg.iMetric$	←	if IsLogicalNode() then SummarizeInstalledMetrics($iMsg_j.iMetric, j \in G_3$) else InstallService($iMsg.servSpec$)

C_0		DS-solicit
S_0	←	SelectAllNodesBetween(<i>iMsg.ends</i>)
$oMsg_n.ends$	←	SelectNeighborNodes($n / n \in S_0$)
G_1	←	S_0



C_1		DS-summarize
S_1	←	GetLogicalNode()
$oMsg.sMetric$	←	if IsLogicalNode() then SummarizeMetrics($oMsg_j.sMetric, j \in G_1$) else CreateMetric($iMsg.servSpec$)
G_2	←	S_1

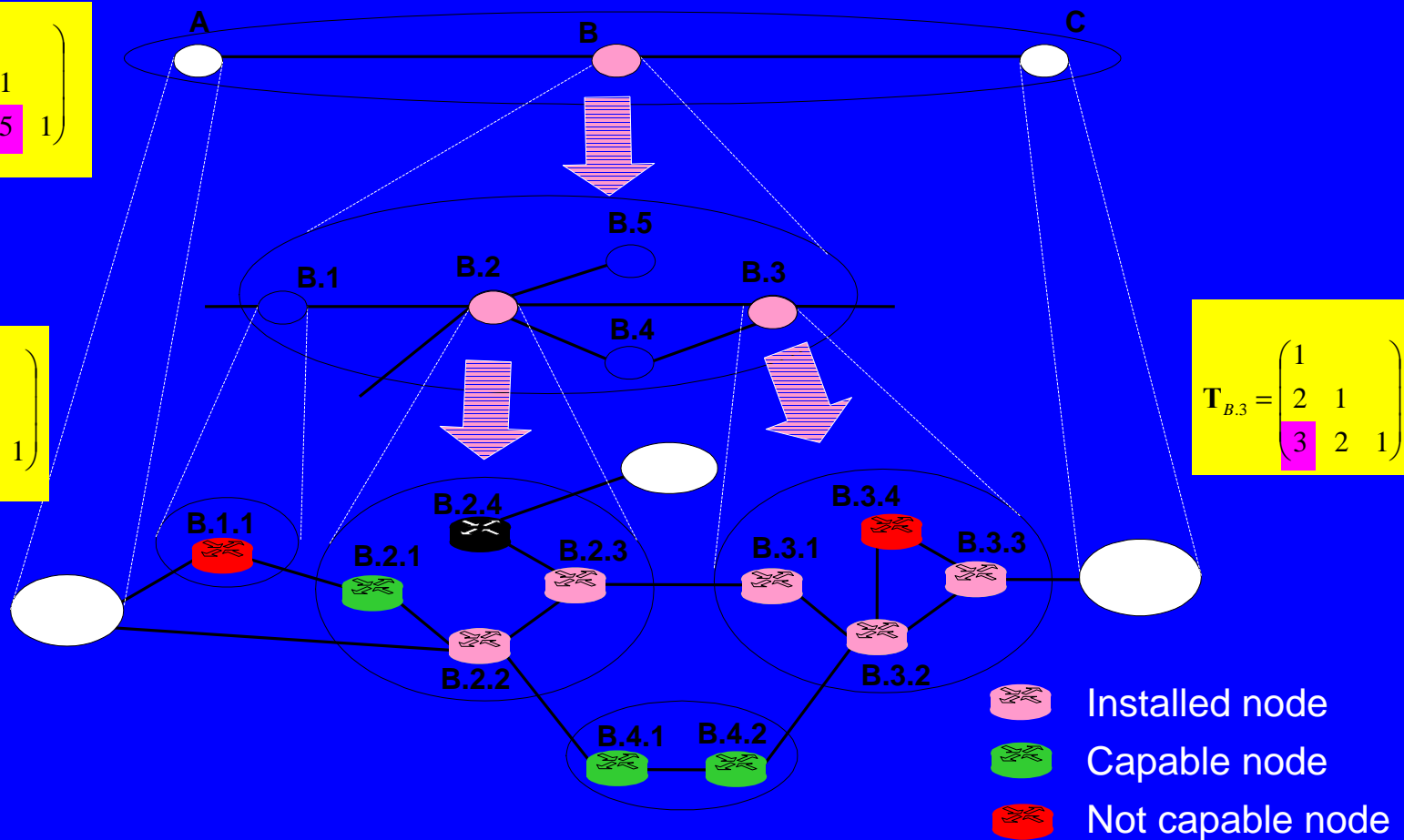


C_2		DS-disseminate
S_2	\leftarrow	if IsLogicalNode() then SelectNodesOnShortestPath(<i>iMsg.ends</i>) else null
$oMsg_n.ends$	\leftarrow	SelectNeighborNodes($n / n \in S_2$)
G_3	\leftarrow	S_2

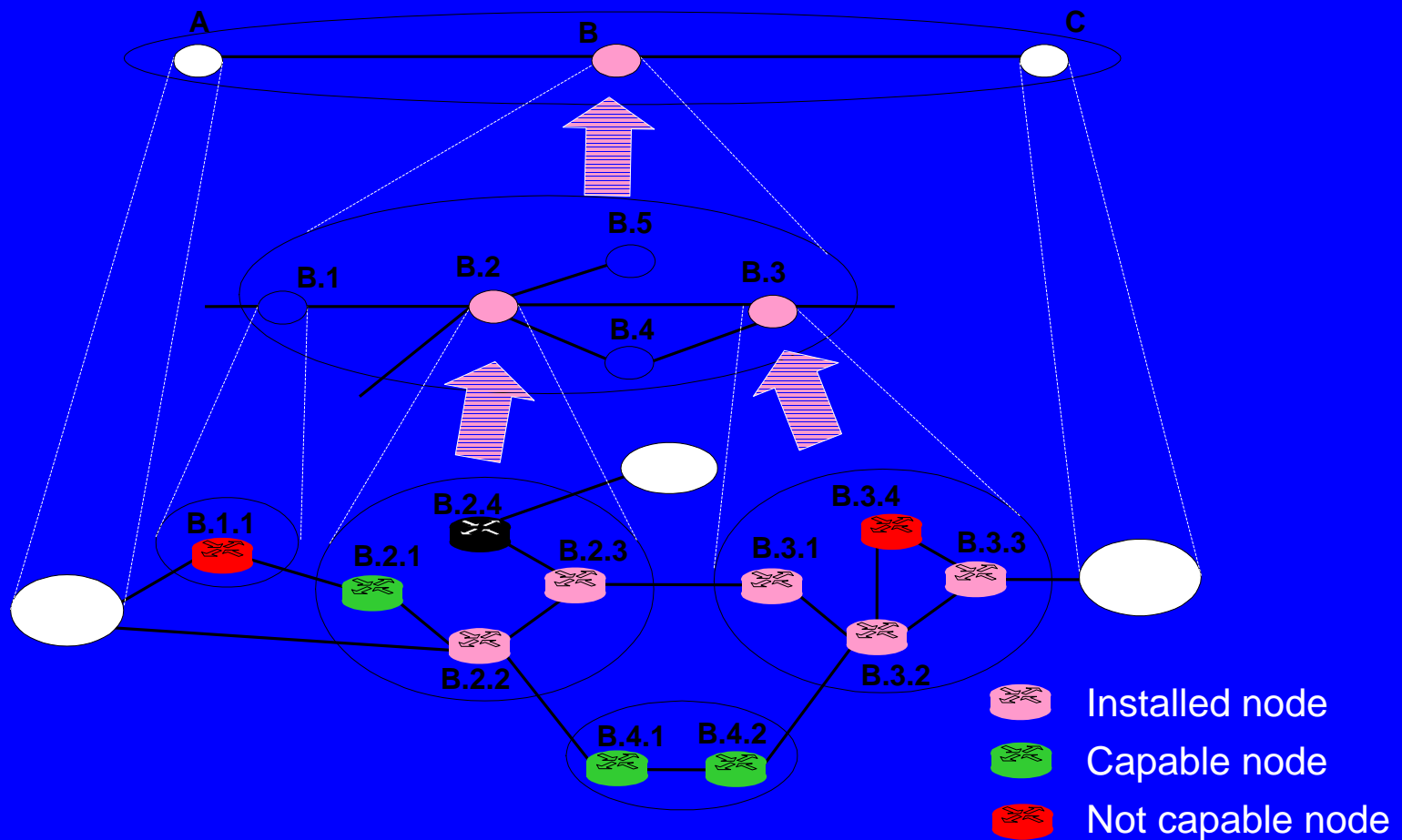
$$\mathbf{T}_B = \begin{pmatrix} 0 & & & & \\ 0 & 1 & & & \\ 0 & 5 & 1 & & \end{pmatrix}$$

$$\mathbf{T}_{B.2} = \begin{pmatrix} 1 & & & & \\ 2 & 1 & & & \\ 3 & 2 & 1 & & \end{pmatrix}$$

$$\mathbf{T}_{B.3} = \begin{pmatrix} 1 & & & & \\ 2 & 1 & & & \\ 3 & 2 & 1 & & \end{pmatrix}$$



C_3		DS-install
S_3	←	GetLogicalNode()
$oMsg.iMetric$	←	if IsLogicalNode() then SummarizeInstalledMetrics($iMsg_j.iMetric, j \in G_3$) else InstallService($iMsg.servSpec$)



Conclusion

- “Internet Services: Management Beyond the Element”
 - Service deployment is an important management function
 - Scalable method for solicitation and collection of information required for proper deployment.
- HIGCS adequate formalization, for example:
 - node-based (web-cache) with $\max(CPU)$ metric
 - path-and-node-based (VPN) with *termination extension for transition matrix*
- Complexity of sparse service deployment
 - Exploration of possible combinations: *ordering rules*
- Challenges for the intelligent network infrastructure:
 - rapidly deploying new services
 - managing heterogeneity (in capabilities and interfaces)
 - supporting yet unpredicted services
 - remain scalable