# 2. Morpheus Broadcast 노드 구현 및 활용

## 프로젝트 개요

Morpheus 파이프라인 내에서 단일 입력 메시지를 여러 개의 다운스트림 스테이지로 동시에 복제 및 전송하는 **Broadcast 노드**의 커스텀 스테이지(MpBroadcastStage)를 구현하고 테스트했습니다. 이는 비선형(Non-Linear) 파이프라인을 구축하는 데 필수적인 기술입니다.

## ⚙️ 사용 기술 및 역할

* **주요 기술:** NVIDIA Morpheus SDK, MRC (Morpheus Runtime Component)
* **핵심 역할:** mrc.core.node.Broadcast를 활용하여 커스텀 스테이지(Stage)를 개발하고, compute\_schema와 \_create\_ports를 오버라이드하여 다중 출력 포트를 정의했습니다.

## 💡 핵심 구현 내용

1. **다중 출력 포트 정의:** 상속받은 Stage 클래스에서 self.\_create\_ports(1, 3)을 호출하여 1개의 입력 포트와 3개의 출력 포트를 명시적으로 정의했습니다.
2. **Broadcast 노드 활용:** \_build 메서드 내에서 mrc.core.node.Broadcast(builder, "broadcast")를 생성하여, 들어오는 메시지를 여러 출력 노드로 복제했습니다.
3. **파이프라인 연결:** Broadcast 스테이지의 각 출력 포트(broadcast.output\_ports[0], [1], [2])를 각각 다른 메모리 싱크 스테이지에 연결하여, 단일 소스 데이터가 세 곳으로 정확히 전달되는 것을 검증했습니다.

## 📌 주요 코드 로직 (MpBroadcastStage 클래스)

import typing  
import mrc.core.operators as ops  
from mrc.core.node import Broadcast  
from morpheus.pipeline import Stage, StageSchema  
# ... 기타 Morpheus, MRC 임포트 ...  
  
# PassThruTypeMixin 대신 Stage를 직접 상속받아 타입을 명시적으로 정의  
class MpBroadcastStage(GpuAndCpuMixin, Stage):  
  
 def \_\_init\_\_(self, c: Config):  
 super().\_\_init\_\_(c)  
 # 1개의 입력 포트와 3개의 출력 포트 정의  
 self.\_create\_ports(1, 3)   
  
 @property  
 def name(self) -> str:  
 return "mp-bcast"  
  
 # 출력 포트의 메시지 타입을 명시적으로 정의 (Broadcast는 복제 메시지를 전송)  
 def compute\_schema(self, schema: StageSchema):  
 for port\_schema in schema.output\_schemas:  
 port\_schema.set\_type(typing.Any) # 또는 MessageMeta, ControlMessage  
  
 def on\_data(self, message: typing.Any) -> typing.Any:  
 # 메시지 내용을 변경 없이 그대로 반환  
 return message  
  
 def \_build(self, builder: mrc.Builder, input\_nodes: list[mrc.SegmentObject]) -> list[mrc.SegmentObject]:  
  
 # 1. Broadcast 노드 생성 및 입력 연결  
 broadcast = Broadcast(builder, "broadcast")  
 builder.make\_edge(input\_nodes[0], broadcast)  
  
 # 2. 3개의 다운스트림 노드 생성  
 outgoing\_1 = builder.make\_node("outgoing\_1", ops.map(self.on\_data))  
 outgoing\_2 = builder.make\_node("outgoing\_2", ops.map(self.on\_data))  
 outgoing\_3 = builder.make\_node("outgoing\_3", ops.map(self.on\_data))  
   
 # 3. Broadcast 노드와 각 출력 노드 연결  
 builder.make\_edge(broadcast, outgoing\_1)  
 builder.make\_edge(broadcast, outgoing\_2)  
 builder.make\_edge(broadcast, outgoing\_3) # Broadcast는 여러 엣지를 가질 수 있음  
  
 # 4. 스테이지의 출력 포트에 대응하는 노드 반환  
 # 참고: self.\_create\_ports(1, 3)으로 3개 포트를 정의했으므로, 3개 노드를 반환해야 합니다.  
 return [outgoing\_1, outgoing\_2, outgoing\_3]