03

오픈 소스로 구현한
22대 국회의원 선거개표 방송시스템

박준헌 ubc 울산방송 미디어기술국 네트워크 팀장

개발 동기

그동안 선거개표 방송을 실시할 때마다 업체들로부터 송출 장비와 Template를 구매 또는 임대하고, 송출될 Template의 디자인도 대부분 업체의 손을 거쳐왔다. 올해는 예산을 긴축한다는 방침으로 별도의 예산 없이 자체 해결하기로 했는데, 데이터를 수동으로 입력해서 특정 편성시간에만 CG를 통해 송출하는 방안으로 결정됐다. 그러나 시시각각 변하는 개표 상황을 한정된 편성시간에만 방송한다는 방안은 지역 시청자를 최우선으로 고려하는 지역방송의 입장에서는 다소 미흡한 방안이라 하겠다. 그렇다고 마땅한 해결 방법이 없이 고민하던 중, 몇 년 전에 조금 공부해 본 CasparCG라는 오픈 소스 소프트웨어가 떠올랐다. 유럽에서는 이미 선거개표방송을 CasparCG로 실시한 바가 있으며, 십수 년 동안 업그레이드를 거듭해서 이제는 충분히 안정되고 검증된 소프트웨어라고 할 수 있다.

그동안 ubc 울산방송은 선거개표방송을 전산실에서 담당해왔으나, 현재는 전산실이 해체되고 그 본연의 업무가 분할돼서 일부 방송 관련 업무와 사내 네트워크 및 WEB 관리를 방송기술 부분으로 이관해서 미디어기술국 네트워크팀에서 수행하게 되었다. 네트워크팀은 RF 업무, 전파행정업무, WEB 서비스 관리, 사내 Network 관리 등 여러 종류의 다양한 업무를 담당하고 있는 터라 많은 시간과 인력이 투입되는 소프트웨어 개발이라는 업무를 병행한다는 것이 쉽지 않은 상황이다. 그렇지만 하나하나 기본 기능들을 구현하고 확인해 가면서 어느 정도 개발에 대한 확신이 들었기에 공식적으로 개발하기로 결정을 내렸다.

애초의 취지가 우리가 현재 가지고 있는 자원만을 가지고 개발하는 것이었고 CasparCG를 다루어 본 경험이 있는 인력은 필자밖에 없는 상황이라 혼자서 개발해야만 했다. 다행스럽게도 나머지 팀원이 통상적인 업무를 흔쾌히 분담하기로 해서 개발에만 집중할 수 있었다. 이러한 난관은 예상했지만 그래도 개발을 추진한 이유는 보수적인 공중파 환경에서 실험적이고 모험적인 시도를 해 볼 기회를 얻기가 매우 어렵기 때문이다. 아이러니하게도 긴축예산이라는 어려운 환경이 새로운 시도를 할 기회를 준 셈이 되었다. 개인적으로는 이 시스템의 개발이 이론적으로 충분히 가능하다는 확신은 있었지만 어디까지나 추측의 영역이고 실증을 한 경험이 없었기에 개발의 성패를 가늠하기 어려웠다. 일단 테스트용 애플리케이션을

작성해서 각각의 기본 개별 기능 요소들을 테스트한 후 전체 시스템으로 통합하는 방식으로 구현하기로 했다. 충분한 개발 인력 없이 한정된 시간에 전체 시스템을 완성하려면 최대한 설계 과정과 구현할 스펙을 줄이고 필수 기능의 구현에만 적절히 시간과 노력을 분배하는 운용의 묘가 필요했다.

CasparCG란 무엇인가?

CasparCG는 스웨덴 방송사인 Sveriges Television(SVT)에 의해 개발된 오픈 소스 그래픽 및 영상 플레이아웃 소프트웨어로서 방송 환경에서 그래픽 오버레이, 애니메이션, 비디오 재생 등을 실시간으로 제공하는 소프트웨어이다.

간단히 설명하자면 주변에서 비교적 쉽게 구할 수 있는 Decklink 편집보드를 이용하여, pixel image file이나, 동영상 클립 또는 HTML 파일, URL 등의 미디어 콘텐츠를 화면에 시현하거나 SDI 신호로 출력해주는 소프트웨어다. 현재 널리 알려진 OBS 프로그램이 Streaming 서비스 용도로 사용되고 있는데, 비슷한 개념으로 CasparCG는 SDI 신호를 출력할 수 있다. 특히 Alpha channel이 적용된 미디어 파일을 송출할 경우에는 CG Key 신호도 출력할 수 있다.

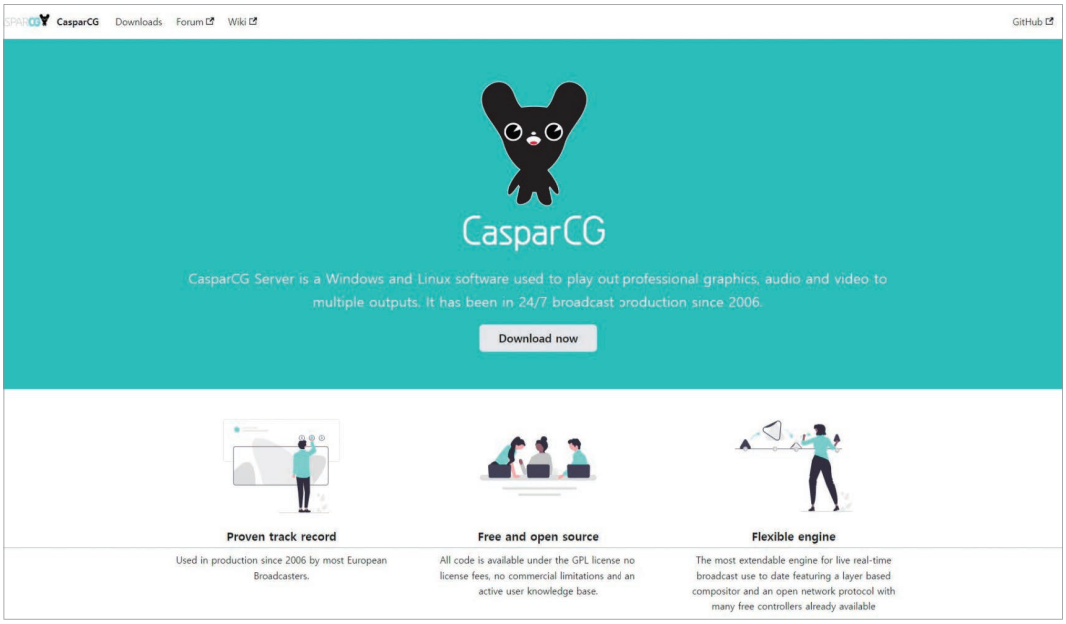

그림 1. CasparCG 홈페이지(casparcg.com)

CasparCG는 크게 CasparCG Server 프로그램, CasparCG Client 프로그램 그리고 미디어 스캐너로 구성된다. CasparCG Server와 CasparCG Client 사이에는 반드시 동일한 소프트웨어 버전으로 맞출 필요는 없다. CasparCG Server와 CasparCG Client 간의 통신은 AMCP(Advanced Media Control Protocol)라는 명령어들로 구성된 프로토콜을 통해서 이루어지는데, 각 명령어와 데이터는 문자열 형식이다. 일반적으로 CasparCG Server는 CasparCG Client로부터 AMCP 명령어를 수신한 후 해당 명령어 또는 데이터를 parsing하고, 명령어와 해당 데이터를 반영하여 실행하며, 그 결과를 영상으로 만들어 SDI 신호로 출력한다.

CasparCG Server는 크게 다음과 같은 내부 구성 요소를 갖추고 있다. 이 구성 요소들은 CasparCG 프로젝트가 시작되고 나서 여러 요구 조건을 반영해 진화해 나가면서 오늘날 형태를 가지게 되었다.

CasparCG의 구성

영상 신호를 포맷에 맞춰 어떤 파일에 저장하든 SDI 신호로 완성하든 생성하기 전에 영상을 만드는 근본이 되는 미디어들과 그 미디어를 재생하는 부분이 있다. CasparCG에서는 이러한 기본 재생 기능을 담당하는 부분을 'Producer'라고 통칭하고 그에 맞춰 생성된 콘텐츠를 SDI 신호나 화면 또는 파일로 출력하는 부분을 'Consumer'라고 한다. 이때 최종 신호를 생성하기에 앞서 Layer 간에 overlay 하거나 Layer별로 Video의 Color 조정 또는 Audio를 Mixer 하거나 volume을 조정하는 기능 등을 담당하는 'Mixer'라고 하는 계층이 둘 사이에 존재한다.

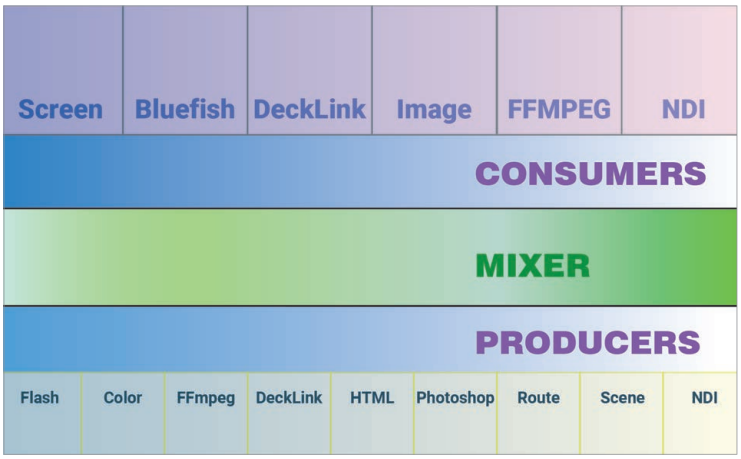

그림 2. producer, consumer, mixer

이러한 기능은 AMCP 내 정의된 개별 명령어들로 각 기능을 실행한다.

ffmpeg producer	ffmpeg이 play 할 수 있는 미디어를 play 한다. QuickTime 비디오 코덱부터 PNG 파일, Motion JPEG, JPEG, H.264, FLV, WMV까지 지원하며, 오디오 코덱 및 비압축 코덱 또한 지원된다.
color producer	색상을 명시해서 matte를 만들어서 출력할 수 있다.
decklink producer	Blackmagic Decklink 카드로부터 입력된 신호를 CasparCG Server의 한 레이어에 올릴 수 있다.
flash producer	Adobe Flash로 만든 swf 파일을 렌더링할 수 있다.
html producer	HTML 페이지를 로드해서 렌더링할 수 있다. 그러나 CasparCG 서버에서 해당 HTML과 interaction 할 수는 없다.
image producer	Alpha 채널이 있거나 또는 없는 pixel image file을 출력할 수 있다.

표 1. Producer 종류와 기능

다음은 주요 AMCP 명령어 리스트와 기능들이다.

play	Clip 또는 이미지 파일 등을 특정 레이어에 play 하는 명령어
stop	해당 레이어에서 동작하던 명령어를 중단하고, 해당 clip을 채널에서 제거한다.
pause	명령어에 동반되는 레이어의 번호에 해당하는 동작을 일시 정지시킨다. resume 명령어로 다시 동작시킨다.
clear	명시된 채널과 레이어에 있던 모든 클립을 제거한다. 레이어를 명시하지 않으면 채널 전체가 삭제된다.

표 2. AMCP 프로토콜 기본 명령어

예를 들어 Media에 Alpha.mp4라는 미디어를 play 하면서 Logo.png라는 alpha channel이 적용된 cg 자막을 overlay 한다면 CasparCG Server로 다음과 같은 명령어를 Socket으로 연속해서 전송하면 된다.

```
"PLAY 1-1 Alpha.mp4"
"PLAY 1-2 Logo.png"
```

Adobe Flash Template와 HTML Template

Template라고 하면 여러 모양의 도형이 뚫려있는 플라스틱판을 쉽게 떠올릴 수 있는데 우리가 사용하는 여러 소프트웨어에서도 자주 사용되는 개념이다. CasparCG에서도 Template는 구성 요소 중 특정 요소를 가변적으로 바꿀 수 있는 문서의 틀이라고 생각하면 이해가 쉬울 것 같다.

CasparCG Template는 프로그래밍적인 요소와 그래픽적인 요소가 뒤섞여 있다. 예를 들어 우리가 개발하고자 하는 개표방송시스템의 템플릿은 Background 이미지 같은 정적인 요소와 후보자 이름, 사진, 득표율, 정당 명칭, 정당 색상 등과 같은 가변적인 요소 그리고 각 요소에 적용될 Animation들로 구성되는데 각각의 요소들은 프로그래밍하는 부분과 그래픽적 요소가 혼재되어 있다. CasparCG에서는 이러한 형식을 다루는 프레임워크로 Adobe의 Flash와 HTML+Javascript를 사용한다.

필자는 7년 전 CasparCG Template 기능을 처음 접할 때 Adobe Flash를 통해서 공부했으나 쉽게 성과를 내지 못했다. 그 이유로는 Flash가 유료 소프트웨어라는 점, 점차 시장에서 사라지고 있는 기술이라는 점, 그리고 무엇보다도 Adobe Flash에서 사용되는 Action Script라는 프로그래밍 언어에 대한 자료가 많지 않아서였다. 필자가 CasparCG를 공부한 시점에서는 아직 HTML Template가 본격적으로 지원되지 않던 시기여서, Template 기능은 개념적인 이해에서 더 이상의 진척이 어려웠다. 그러나 그동안 CasparCG는 많은 업그레이드를 통해서 개량되었고 HTML Template에서는 현재 많이 사용되고 있는 WEB FrontEnd Framework들을 그대로 사용할 수 있게 되었다. CasparCG를 이용한다면 오늘날 널리 사용되고 있는 WEB FrontEnd 기술로 표현되는 콘텐츠를 SDI 신호로 내보낼 수 있다.

템플릿 디자인

ubc 선거개표 방송시스템에서 사용되는 디자인은 표현 방식에 따라 상단, 전체, 하단 템플릿으로 나눈다. 보통 CG실에서 제작한 템플릿의 원형은 psd 파일로 주어지는데 개별 layer들의 그래픽 요소들을 판단하고 변화가 없는 Static 한 요소들은 layer를 합친다. 가변적인 요소들은 데이터가 입혀질 요소들과 Animation이 적용될 요소들을 하나하나 개별 요소로 분리해서 layer를 만든다. 이 Layout이 그대로 유지되도록 HTML, CSS 파일 상에서 Element를 구성하고, CSS 값을 조정한다. 이 과정에서는 편리한 고성능의 HTML 에디터 툴에 의존해서 Layout을 HTML로 작성하지 않는 것이 좋다. 왜냐하면 일반적으로 HTML 에디터 툴들은 완벽하게 디자인된 그대로 HTML로 만들지는 몰라도, 프로그래머가 이해하기 힘들고 편집하기 어려운 복잡한 형태의 CSS 파일을 생성하기 때문이다. 가능하면 가독성이 좋고, 수정하기 좋은 최대한 단순한 형태로 코드를 작성하는 것이 차후 작업을 위해서 바람직하다.

기존의 CG 장비들은 TGA 포맷 이미지들을 기본적으로 사용하지만, WEB 기반의 Template인 관계로 jpg 또는 png 포맷의 이미지들을 사용하게 된다. 특히 Alpha channel을 빈번하게 사용하고 CSS에서 Mask를 지정해서 Animation을 구사하므로 png 파일이 중요한 역할을 한다.

Template를 만들기 위해서는 먼저 CG실에서 만들어준 디자인에서 크게 정적 이미지 부분, 가변적 이미지 부분과 그에 연관되는 이름, 숫자 값 등 문자열 부분 그리고 각 요소에 적용될 Animation 부분으로, 개념적으로 나눈다. 개별 Animation이 적용될 요소를 기준으로 Element의 ID를 할당하고, 그룹으로 묶어서 Animation을 적용할 경우에는 필요에 따라 Element에 Class를 할당한다. 그다음 숫자, 문자, 기호가 포함된 문자열을 기준으로 분류하여 해당 ID와 Class를 할당한다. HTML 내에서 제어할 특정의 Element는 ID 속성을 통해 구별되므로 가능하면 용도에 따라서 접두문자열을 구별해서 이름 짓는 것이 디버깅이나 빠른 코드 작성을 도움이 된다.

전체 디자인에서 레이아웃, 폰트 선정, 배경 이미지 등은 미리 결정되지만, 개별 Animation은 Javascript와 CSS로 제어하고, 하나씩 실행을 시켜가면서 DevTool로 디버그를 해야 하므로 많은 시간과 노력이 소요된다. 특히 Animation의 느낌을 직접 테스트해보고, 원하는 효과에 맞도록 코드를 최적화하는 과정은 시행착오의 연속이라 많은 인내심과 긴 시간이 소요되기 때문에 실제 완성된 Animation의 실행 duration이 무척 짧다는 점을 생각한다면 허무하다는 느낌마저 든다.

시스템 전체 구성

CasparCG Server를 이용해서 SDI 신호를 생성한다는 기본 기능이 달성되었다면, 시스템의 나머지는 선거개표데이터를 어떻게 가져오는가의 문제로 귀결된다. 이전 선거방송과 마찬가지로 올해의 선거방송 데이터 또한 SBS가 제공하는 선거개표 데이터를 사용하기로 했다. 그런데 이전 선거방송부터 SBS 선거개표데이터 수신용으로 사용해 온 MS-SQL 서버의 하드웨어와 운영체계가 너무 낡아서 금방이라도 고장날 것 같았다. 애초에 SDI 생성 부분에만 집중하고 기존 MS-SQL 서버를 그대로 이어받아 시스템에 포함하려 했지만, 언제 고장이 나도 이상하지 않을 만큼 구형 장비라 다른 대안을 찾기로 했다.

이번 선거에서 울산광역시는 총 여섯 군데의 선거구가 해당하므로 고성능 Database는 필요 없었기에 가상화 서버 container를 주, 예비 2기 생성하고 무료 오픈 소스 RDB인 MariaDB를 올려서 사용하기로 했다. 물론 그에 따라서 기존 MS-SQL에 기반해 만들어진 SBS에서 개표데이터 수신용 Data Copy Application은 더 이상 사용할 수 없게 되었다.
SBS에서 제공하는 선거개표 데이터는 Amazon Web Service(AWS) 상에서 구동하는 IBM DB2 데이터베이스를 통해서 얻어온다. 필자는 DB2를 사용해 본 적이 없어서, AWS에서 MariaDB로 데이터를 가져올 Data Copy Application을 어떻게 만들지 고민했는데, 촉박한 일정으로 python과 ibm_db 모듈을 이용해서 몇 개의 기초적인 로그 기능과 최소한의 데이터 저장 및 업데이트 기능만 구현한 Application을 작성해서 사용했다.

이제 남은 기능은 수동선거 데이터 입력기능과 Sequence 송출 기능이다. 수동으로 선거개표 데이터를 입력하는 기능은 인터넷이 끊어져서 AWS에 접속이 불가능해 개표데이터를 못 가져올 경우를 대비해 직접 MariaDB에 데이터를 입력하는 기능이다. 이 파트는 현재 각 선거구의 개표율, 후보의 득표율 순위 등

개표 현황을 보여주는 Viewer 역할도 하고 동시에 수동으로 데이터를 입력하는 기능도 함께 하도록 작성했다.

Sequence 송출 기능은 생방송에서 사용자가 작성한 큐시트대로 자동으로 해당 Template를 송출하는 Sequence 송출 제어 기능이다. 이 기능을 세부적으로 보면 MariaDB로 부터 개표데이터를 가져와서 필요한 정보 형식으로 데이터를 생성한 후 특정 시점에서 CasparCG Server로 데이터를 전송하기도 하고 송출 순서에 따라 CasparCG Server로 지정된 Template를 play 하도록 명령어를 내리는 기능이다.

해당 프로그램의 제작은 C#으로 작성했고, Sequence 송출 기능은 일반화된 체계를 설계한 후에 클래스 기반으로 차근차근 구현하는 것이 원칙이지만 시간이 촉박하여 최대한 단순 기능을 구현한다는 느낌으로 프로그램을 제작해야 했다. 특히 Sequence 송출 기능을 체계 설계 후 Thread를 기반으로 구현하려고 하니 디버깅이 어렵고 많은 시간이 소요할 것으로 보여 Thread 대신 Timer를 사용했다. Sequence 송출 기능의 핵심은 일회성 실행인지 반복 실행인지 사용자의 선택에 따라, Template가 반복 횟수를 조절할 수 있어야 한다는 점이다. 이러한 기능의 애플리케이션은 유한상태머신(FSM)의 형태를 띠게 되는데, Class를 설계해서 구현하지 않고 간략하게 Timer Event Handler에서 명령어를 보내는 메서드를 호출하는 형식으로 구현하고 말았다. 그렇지만 Thread를 사용하는 것보다 디버깅하기 쉬워서 시간상 이점으로 작용했다.

전체 시스템 동작 원리

전체 시스템의 동작 원리를 정리하자면 다음 그림과 같다. SBS 선거개표데이터가 AWS DB2 서버에 올라가면 15초에 1회씩 Data Copy Application이 울산에 소속된 선거구 데이터를 읽어오고, Update 용 Table과 Event 저장 Table에 각각 저장한다.

Sequence 프로그램은 Update 용 Table에 저장된 자료를 읽어 와서 스케줄링된 Template에 맞도록 정보를 재가공해서 Timing에 맞게 CasparCG Server로 명령어 및 Data를 전송하고 CasparCG Server는 AMCP로 입력된 명령어와 데이터 템플릿 이름을 parsing 해서 명령어에 맞게 동작한다. 동시에 CasparCG Server의 출력단으로 SDI 신호가 나오고 이 신호는 제작 Switcher로 입력되어 Key 신호 또는 전체 Background로 송출된다.

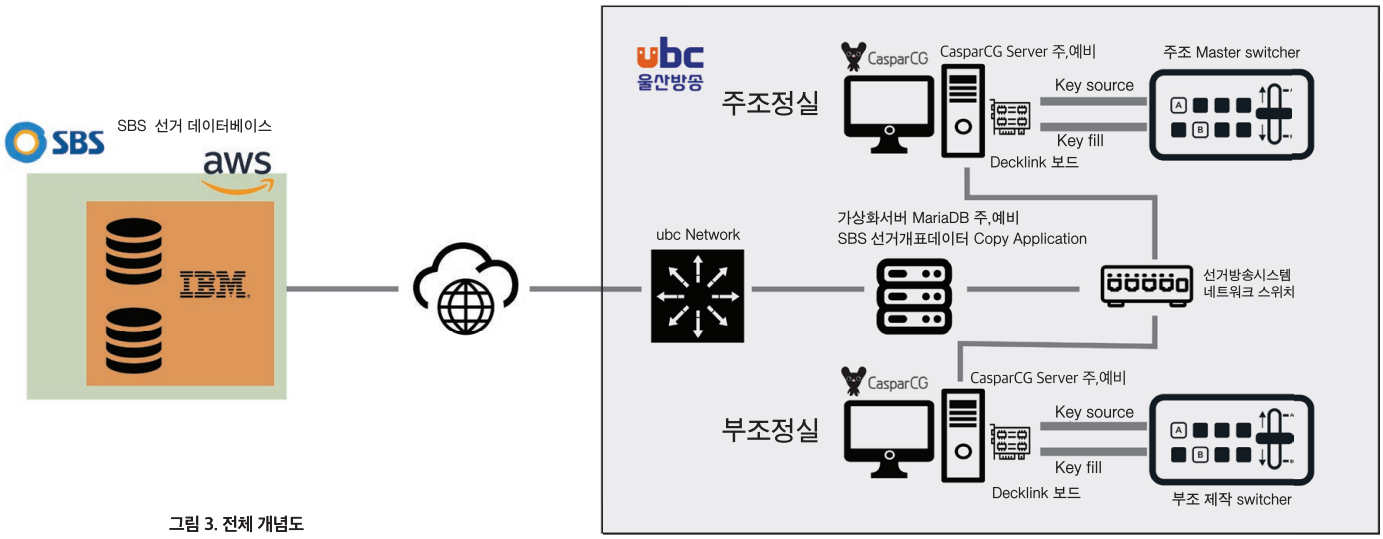

그림 3. 전체 개념도

마지막 리허설 그리고 생방송

선거일을 일주일 앞두고 시스템 시연회를 열었는데, 이 자리에서는 참석자 의견 등을 청취하고 추가로 구현할 기능들에 대한 요구를 수렴했다. 막상 시연회에서 별다른 의견이 나오지 않았으나 선거 전날 최종 리허설을 진행했을 때는 아나운서와 PD로부터 필요한 요구 사항들이 쏟아졌다.

스탠바이 기능과 아나운서의 멘트에 맞춰 다음 화면으로 넘기는 기능이 추가로 필요했다. 프로그램을 수정할 수 있는 시간은 고작 하루밖에 없는 상황. 해당 기능의 구현은 조작을 담당하는 인터페이스부터 내부 로직까지 통째로 수정해야 하는 작업이어서 새벽까지 수정과 디버깅을 반복해야 했다. 애초에 부조정실과 주조정실에만 설치하기로 했는데, 긴급하게 Youtube 송출을 담당하는 뉴미디어팀에서도 추가로 시스템을 요구했다. 남아있는 PC에 여분의 Decklink 보드를 설치한 후에 긴급하게 프로그램을 설치해서 배포했다. 급하게 요청을 받아도 즉각 대응할 수 있다는 점은 자체적으로 시스템 제작을 할 수 있었기에 가능한 매우 큰 장점이다.

드디어 생방송이 시작되었다. preview를 통해서 key를 확인하고 송출하려는데 미처 생각하지 못했던 문제가 발생했다. 선거구마다 개표 상황이 달라서 어떤 선거구는 0%의 개표율이라 송출리스트에서 빼야 하는데 이런 상황을 예상하지 못한 것이다. 또 다른 문제로는 하단 Template에서 기존 방송의 하단을 덮을 정도로 사이즈를 키웠음에도 수화가 송출되는 부분의 크기를 고려하지 못해서 긴급하게 템플릿을 수정해야만 했다. 어렵게 디자인한 그래픽을 임시방편으로 수정해야만 했다.

가지고 있던 부품들로 총 5기의 CasparCG Server를 만들어서 사용했는데, 워낙 단순하게 Application을 제작했던 터라 사용상 크게 어려운 점은 없었고, 선거개표방송이 종료될 때까지 별다른 어려움 없이 사용했다.

막상 생방송을 마치고 나서야 비로소 시스템의 설계 방향에 대한 감이 잡혔다고 한다면 이상하게 들리겠지만 단순히 On Air로 화면을 송출하는 기능만 고려해서 만든 최소한의 System이기에 생방송을 끝낸 이후에야 수정할 많은 결함들이 보였다. 운영상의 편의성, 예상하지 못했던 상황에 유연하게 대처할 수 있는 레이아웃 디자인, 그래픽의 수준도 중요하지만, 어떤 지표, 자료를 보여줄 것인가라는 점, 그리고 하나의 Template를 어려운 Animation을 적용해 공들여 제작하기보다는 단순한 형태의 템플릿들을 다양하게 만들어 지루하지 않도록 하는 것이 더 선호된다는 점 등은 실전을 통해서만 확인할 수 있는 수확물이었다.

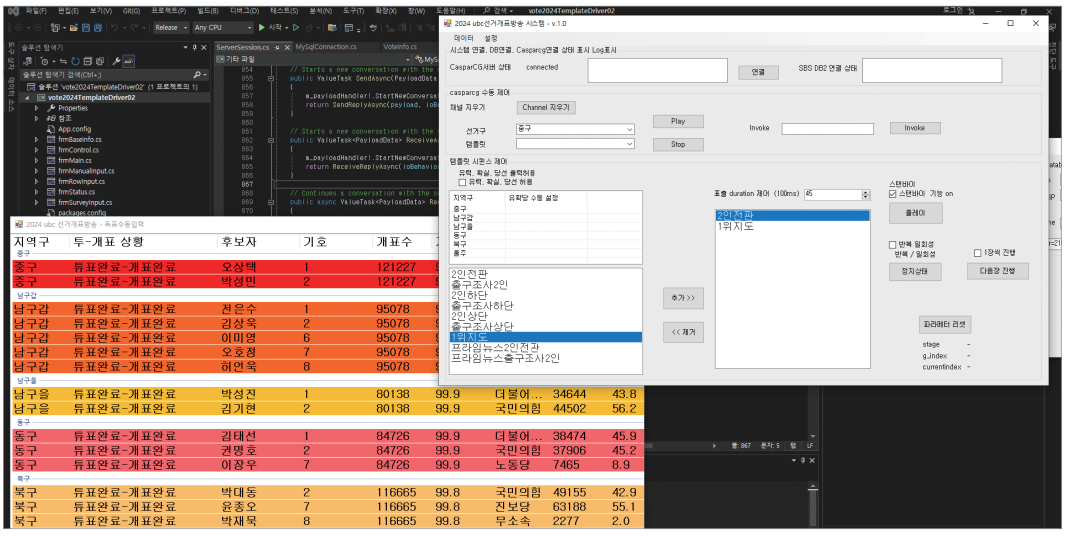

그림 4. 애플리케이션

그림 5. 전체 Template 송출

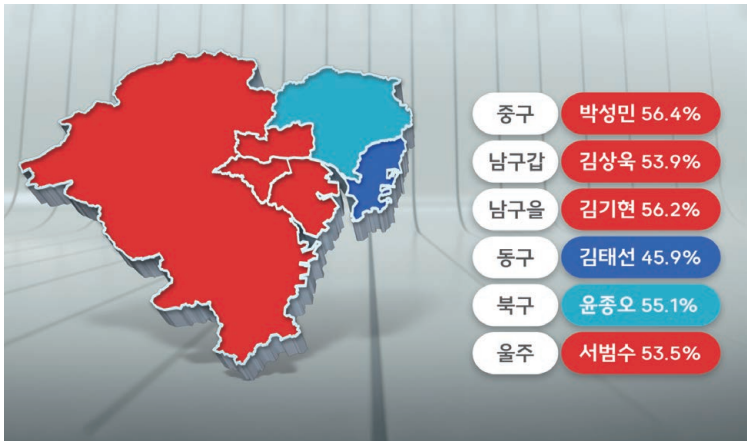

그림 6. 지도 Template

에필로그

필자의 기억 속에는 초등학교 시절 전파상 쇼윈도에서 아날로그 컬러TV를 처음 보고 신기해했던 기억과 중년의 나이에 아날로그 TV 마지막 전파를 송출하던 송신기를 직접 껐던 순간이 겹쳐 있다. 그 후 UHDTV 방송국을 개국하던 기억을 돌이켜 보면 방송기술의 발전 속도는 불현듯 급속도로 빨라진 듯하다. 어린이가 중년이 될 때까지 유지되던 방송 방식이 폐지되고 벌써 두 번이나 새로운 방송규격이 나왔으니 말이다. 그러나 정보통신기술 전반의 발전은 이 속도를 아득히 뛰어넘는다. '인터넷과 WEB의 시대', '스마트폰의 시대', '빅데이터의 시대'를 거쳐, 이제는 '생성 AI'의 시대에 진입하고 있다. '급변하는 방송환경'이라는 수식어는 더 이상 마음속에서 감흥을 불러일으키지 않는다. 방송기술시장의 무게중심도 '브로드캐스트'에서 급격히 '1인 미디어'로 급격히 옮겨가고 있다.
만성적인 인원 부족, 맥락 없이 산재하고 파편화된 업무들, 방송업계 전반의 부진, 뉴미디어의 등장, 신기술의 습득 요구. 이러한 일들은 서로 무관한 이벤트가 아니라 같은 원인에서 출발한 다른 현상이다. 가장 근본적인 원인은 국경도 없고 규제도 없는 대규모 자본을 가진 거대 플랫폼과 기술 경쟁에 직면하고 있는 현실 때문일 것이다.

필자는 이러한 험난한 방송환경을 헤쳐나가는데 프로그래밍 능력이 방송기술자에게는 유용하고 필수적인 기술이 될 것으로 전망해본다. 오늘날 생산되는 방송장비는 Automation 기능, Network 연결, WEB 설정 같은 기능이 기본으로 탑재되며, 이런 기능은 최대한 사용하려면 프로그래밍 능력이 많은 도움이 된다. 전통적인 관점의 연결이 물리적 케이블과 납땜으로 상징된다면 새로운 연결의 의미는 네트워킹, API 호출과 같은 IT 기술에 기반을 둔다.
소프트웨어의 시대인 지금 프로그램을 작성하는 능력은 오랜 시간 전문교육을 거친 인력들의 전유물도 더 이상 아니다. 무료로 사용할 수 있고 쉽게 다룰 수 있는 컴파일러가 많이 있고, 어렵지 않게 여러 예제 코드를 인터넷으로 검색하고 참고할 수 있으며, 더욱이 ChatGPT와 같은 인공지능의 도움을 받으면 생산성을 획기적으로 올릴 수도 있다. 프로그래밍 능력을 갖춘 방송 엔지니어에게는 전통적인 방송기술의 영역 넘어 더 크고 넓은 기회의 영역이 열려 있다.

개발하는 과정에서 세세하게 고려하지 못했던 서버, 네트워크 구축을 맡아서 많은 도움을 준 윤기원 부장, 박기윤 사원과 빠른 시간에 그래픽 시안을 완성해 준 송정근 CG실장에게 본 지면을 빌어 개인적으로 감사의 인사를 전한다. 그리고 다소 도전적이고 모험적인 프로젝트를 허락해 준 기술국장님과 보도국장님께 깊은 감사의 인사를 드린다.